# FINANCIAL FORECASTING USING EVOLUTIONARY COMPUTATIONAL TECHNIQUES

A THESIS SUBMITTED IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & INSTRUMENTATION ENGINEERING.

By

Adwait Jog
Roll No: 10507004
And
Avijit Mohapatra
Roll No: 10507003


**Department of Electronics & Communication Engineering**
**National Institute of Technology, Rourkela**

# FINANCIAL FORECASTING USING EVOLUTIONARY COMPUTATIONAL TECHNIQUES

A THESIS SUBMITTED IN
PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & INSTRUMENTATION ENGINEERING.

By

Adwait Jog
Roll No: 10507004
And
Avijit Mohapatra
Roll No: 10507003


**Department of Electronics & Communication Engineering**
**National Institute of Technology, Rourkela**

National Institute of Technology
Rourkela

# CERTIFICATE

This is to certify that the thesis **"Financial forecasting using evolutionary computational techniques**" submitted by **Adwait Jog** and **Avijit Mohapatra** in partial fulfillment of requirements for the award of Bachelors in Technology degree in Electronics and Instrumentation Engineering, Department of Electronics and Communication Engineering at National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any university /institute for award of any degree or diploma.

Date : 10th May 2009

Dr. Ganapati Panda

Professor

Department of ECE

National Institute of Technology

Rourkela 769008

# ACKNOWLEDGEMENT

Our heart pulsates with the thrill for tendering gratitude to those persons who helped us in completion of the project. The most pleasant point of presenting a thesis is the opportunity to thank those who have contributed to it. Unfortunately, the list of expressions of thank no matter how extensive is always incomplete and inadequate. Indeed this page of acknowledgment shall never be able to touch the horizon of generosity of those who tendered their help to us.

First and foremost, we would like to express our gratitude and indebtedness to Dr. Ganapati Panda, for his kindness in allowing us for introducing the present topic and for his inspiring guidance, constructive criticism and valuable suggestion throughout this project work. We are sincerely thankful to him for his able guidance and pain taking effort in improving my understanding of this project.

We are also grateful to Prof. S.K. Patra (Head of the Department) for assigning us this interesting project and for his valuable suggestions and encouragements at various stages of the work. We are also grateful to Ms. Babita Majhi and Mr. Debidutta Mohanty for their valued suggestions and inputs during the course of the project work.

An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. I acknowledge my indebtedness to all of them.

Last but not least, my sincere thanks to all my friends who have patiently extended all sorts of help for accomplishing this undertaking.

Date: 10th May 2009
Place: NIT Rourkela

Avijit Mohapatra
Dept. of ECE Engineering
National Institute of Technology
Rourkela – 769008

Adwait Jog
Dept. of ECE Engineering
National Institute of Technology
Rourkela – 769008

# ABSTRACT

Financial forecasting or specially stock market prediction is one of the hottest field of research lately due to its commercial applications owing to high stakes and the kinds of attractive benefits that it has to offer. In this project we have analyzed various evolutionary computation algorithms for forecasting of financial data. The financial data has been taken from a large database and has been based on the stock prices in leading stock exchanges .We have based our models on data taken from Bombay Stock Exchange (BSE), S&P500 (Standard and Poor's) and Dow Jones Industrial Average (DJIA). We have designed three models and compared those using historical data from the three stock exchanges. The models used were based on:

1. Radial Basis Function parameters updated by Particle swarm optimization.

2. Radial Basis Function parameters updated by Least Mean Square Algorithm.

3. FLANN parameters updated by Particle Swarm optimization.

The raw input for the experiment is the historical daily open, close, high, low and volume of the concerned index. However the actual input to the model was the parameters derived from these data. The results of the experiment have been depicted with the aid of suitable curves where a comparative analysis of the various models is done on the basis on various parameters including error convergence and the Mean Average Percentage Error (MAPE).

**Key Words: Radial Basis Functions, FLANN, PSO, LMS.**

## TABLE OF CONTENTS

# Chapter 1

# INTRODUCTION

Introduction to Stock Market Prediction

Application of Statistical and Soft Computing

Techniques to Financial Forecasting

## 1.1 Introduction to Stock Market Prediction

Financial Forecasting or specifically Stock Market prediction is one of the hottest fields of research lately due to its commercial applications owing to the high stakes and the kinds of attractive benefits that it has to offer. Forecasting the price movements in stock markets has been a major challenge for common investors, businesses, brokers and speculators. As more and more money is being invested the investors get anxious of the future trends of the stock prices in the market. The primary area of concern is to determine the appropriate time to buy, hold or sell. In their quest to forecast, the investors assume that the future trends in the stock market are based at least in part on present and past events and data **[1].** However financial time-series is one of the most 'noisiest' and 'non-stationary' signals present and hence very difficult to forecast **[2][3]**.

The Dow Jones Industrial Average (DJIA) index was launched in 1896 with 12 stocks and is now the worlds most often quoted stock exchange index, based on a price-weighted average of 30 significant companies traded in the New York Stock Exchange (NYSE) and NASDAQ. The index gives a general indication of the behavior of the market towards different information. The other well known index, considered by researchers for prediction, are the Standard & Poor (S&P) 500 and the Bombay Stock Exchange. Many researchers in the past have applied various statistical and soft computing techniques such as neural networks to predict the movements in these stock indices. Generally technical indicators like moving averages and relative strength indices derived from the time series of these indices is employed in this regard.

Financial time-series has high volatility and the time-series changes with time. In

addition, stock market's movements are affected by many macro-economical factors such as political events, firms' policies, general economic conditions, investors' expectations, institutional investors' choices, movement of other stock market, psychology of investors, etc **[4]**. Nevertheless there has been a lot of research in the field of stock market prediction across the globe on numerous stock exchanges; still it remains to be a big question whether stock markets can really be predicted and the numerous challenges that exist in its everyday application on the stock floor by the institutional investors to maximize returns. Generally there are three schools of thoughts regarding such prediction. The first school believes that no investor can achieve above average trading advantages based on historical and present information. The major theories include the Random Walk Hypothesis and the Efficient Market Hypothesis **[5] [6]**.The second view is that of Fundamental Analysis. Analysts undertake in depth studies into the various macro-economic factors and look into the financial conditions and results of the industry concerned to discover the extent of correlation that may exist with the changes in the stock prices. Technical Analysis presents the third view on market price prediction. Analysts attempt to extract trends in market using past stock prices and volume information. These trends give insight into the direction taken by the stock prices which help in prediction. Technical Analysts believe that there are recurring patterns in the market behavior, which can be identified and predicted. In the process they use number of statistical parameters called Technical Indicators and chart patterns from historical data.

## 1.2 Application of Statistical and Soft Computing Techniques to Financial Forecasting

As the underlying theory behind all these techniques is totally different they generally give quite contradictory results. More importantly, these analytical tools are heavily dependent on human expertise and justification in areas like, the location of reversal (or continuation) pattern, market pattern, and trend prediction. For such reasons researchers have stressed on developing models for accurate prediction based on various statistical and soft computing techniques. One such statistical technique employed in this regard is the Auto Regressive Integrated Moving Average (ARIMA) based model. Different time-series in practice have different frequency components. However, there is no systematic approach or a suitable class of models available in the literature to accommodate, analyze and forecast time-series with changing frequency behavior via a direct method. The virtue of ARIMA (Auto Regressive Integrated Moving Average) is well characterized by Vandaele: "… can be viewed as an approach by which time series data sifted through a series of progressively finer sieves…" The aim of sifting some components is to identify so called "white-noise-processes" which has merely stochastic influences on the time series. The recent advancement in soft computing has given new dimension to the field of financial forecasting. Tools based on ANN have increasingly gained popularity due to their inherent capabilities to approximate any nonlinear function to a high degree of accuracy. Neural networks are less sensitive to error term assumptions and they can tolerate noise, chaotic components **[7]**. Banks and Financial Institutions are investing heavily in development of neural network models and have started to deploy it in the financial trading arena. Its ability to 'learn' from the past and produce a generalized model to forecast future prices, freedom to incorporate fundamental

and technical analysis into a forecasting model and ability to adapt according to the market conditions are some of the main reasons for its popularity. Radial Basis Function (RBF) **[8]**, Recurrent Neural Network (RNN) **[9]** and Backpropagation in Multilayer Perceptron (MLP) are the three most popular Artificial Neural Network (ANN) tool for the task. On top of these, evolutionary approaches such as Genetic Algorithm (GA) **[10]**, confluence of statistics and ANN, are receiving attention as well.

# Chapter 2

## A Survey of Existing ANN Models for Stock Market Prediction

A lot of research has gone into the development of models based on a range of intelligent soft computing techniques over the last two decades. Early models employed the Multi Layer Perceptron (MLP) architecture using Backpropagation algorithm, while a lot of recent work is based on evolutionary optimization techniques such as Genetic Algorithms (GA). This section describes briefly some of work that has gone into the field of application of ANN to stock price prediction.

In Japan, technology major Fujitsu and investment company, Nikko Securities joined hands to develop a stock market prediction system for TOPIX, the Tokyo based stock index, using modular neural network architecture **[14]**. Various economic and technical parameters were taken as input to the modular neural network consisting of multiple MLP used in parallel.

A study was done on the effect of change of network parameters of an ANN Backpropagation model on the stock price prediction problem **[15]**. The paper gives insights into the role of the learning rate, momentum, activation function and the number of hidden neurons to the prediction.

In addition to ANN using Backpropagation, the Probabilistic Neural Network (PNN) has also been employed to stock prediction **[16]**. In their work, the model is used to draw up a conservative thirty day stock price prediction of a specific stock:

Apple Computers Inc. Due to their bulky nature owing to the large training data, the PNN are not popular among forecasters.

In the process lots of newer architectures came to the fore. (Ornes & Sklansky) **[17]** in their paper present a Visual Neural Network (VNN), which combines the ability of multi expert networks to give low prediction error rates with visual explanatory power of nonlinear dimensionality reduction. They conclude that the VNN is a powerful means of interactive neural network design, which provides both better prediction accuracy and good visual explanatory ability.

Another architecture introduced to the prediction problem is the Multi Branch Neural Network (MBNN) proposed by (Yamshita, Hirasawa & Hu, 2005) **[18]** and applied to the TOPIX (Tokyo Stock Exchange). The simulations show that MBNN, based on the concept of Universal Learning Networks (ULN), have higher accuracy of prediction than conventional NNs .

In their paper, (Chen, Dong & Zhao, 2005) **[19]** investigate how the seemingly chaotic behavior of stock market could be well represented using Local Linear Wavelet Neural Network (LLWNN) technique. They considered the NASDAQ-100 index and S&P CNX NIFTY index (India). The LLWNN is optimized by using Estimation of Distribution Algorithm (EDA).Results show that the LLWNN model performs marginally better than conventional NN models. Hybrid architectures are also being deployed in recent times. (Raymond Lee, 2004) **[20]** proposed a Hybrid Radial Basis Function Recurrent Network (HRBFN) stock prediction system called the iJADE stock advisor. The stock advisor was applied to major Hong Kong stocks and produced promising results in terms of efficiency, accuracy and mobility.

Another Hybrid AI approach to the implementation of trading strategies in the S&P 500 index futures market is proposed by (Tsiah, Hsu & Lai,) **[21]**. The Hybrid AI approach integrates the rule-based systems techniques with Reasoning Neural Networks (RN) to highlight the advantages and overcome the limitations of both the techniques. They demonstrate that the integrated futures trading system (IFTS) based on this hybrid model outperforms other conventional NN.

There are instances of application of fuzzy logic based models to the stock market prediction as well. Hiemstra proposes a fuzzy logic forecast support system to predict the stock prices using parameters such as inflation, GNP growth, interest rate trends and market valuations **[22]**. According to the paper, the potential benefits of a fuzzy logic forecast support are better decision making due to the model-based approach, knowledge management and knowledge accumulation.

Another effort towards the development of fuzzy models for stock markets has been made by (Alaa Sheta, 2006) **[23]** using Takagi-Sugeno (TS) fuzzy models. Sheta uses the model for two non-linear processes, one pertaining to NASA and the other to prediction of next week S&P 500 index levels. The two steps involved in the process are 1) the determination of the membership functions in the rule antecedents using the model input data; 2) the estimation of the consequence parameters. Parameters are estimated using least square estimation.

The application of evolutionary optimization techniques such as Genetic Algorithm has given an entirely new dimension to the field of stock market prediction. (Badawy, Abdelazim & Darwish) **[24]** conducted simulations using GA to find the optimal combination of technical parameters to predict Egyptian stocks accurately.

(Tan, Quek & Ng, 2005) **[25]** introduced a novel technique known as Genetic Complementary Learning (GCL) to stock market prediction and give comparisons to demonstrate the superior performance of the method. GCL algorithm is a confluence of GA and hippocampal complementary learning.

Another paper introducing Genetic algorithm approach to instance selection (GAIS) (Kyoungjae-Kim, 2006) **[26]** for ANN in financial data mining has been reported. Kim introduces this technique to select effective training instances out a large training data set to ensure efficient and fast training for stock market prediction networks. The GA also evolves the weights that mitigate the well known limitations of the gradient descent algorithm. The study demonstrates enhances prediction performance at reduced training time.

A hybrid model proposed by (Kuo, Chen & Hwang, 2001) **[27]** integrates GA based fuzzy logic and ANN. The model involves both quantitative factors (technical parameters) and qualitative factors such as political and psychological factors. Evaluation results indicate that the neural network considering both the quantitative and qualitative factors excels the neural network considering only the quantitative factors both in the clarity of buying-selling points and buying and selling performance.

Another hybrid model involving GA proposed by (Hassan, Nath & Kirley, 2006) **[28]** utilizes the strengths of Hidden Markov Models (HMM), ANN and GA to forecast financial market behavior. Using ANN, the daily stock prices are transformed to independent sets of values that become input to HMM. The job of the GA is to optimize the initial parameters of HMM. The trained HMM is then used to identify and locate similar patterns in the historical data.

A similar study investigates the effectiveness of a hybrid approach based on Time Delay Neural Networks (TDNN) and GA (Kim & Shin, 2006) **[29]**. The GA is used to optimize the number of time delays in the neural network to obtain the optimum prediction performance.

Other studies and research in the field of stock market prediction using soft computing techniques include comparative investigation of both the ANN and the statistical ARIMA model (Schumann & Lohrbach, 1994) **[30]** for the German stock index (DAX).The ANN method uses the four layer counter propagation network. The paper compares the results provided by both the methods and concludes that the efficient market hypothesis does not hold good. A Data Compression Techniques for stock prediction (Azhar, Badros & Glodjo, 1994) **[31]** has been reported that uses the vector quantization method as an example of lossy data compression and Lempel-Ziv method as an example of lossless data compression technique to predict most of the well known indices across the globe.

# Chapter 3

## INTRODUCTION TO THE MODELS

Introduction to Function Linked Artificial Neural Network based model
Structure of FLANN
Learning of FLANN network
Introduction to Radial Basis Function based model
Structure of RBF
Learning methods - LMS & PSO

## 3.1 Introduction to FLANN based model for stock price prediction

This study proposes a Functional Link or FLANN architecture based model to predict the movements of prices in the DJIA and S&P500 stock indices. The functional link ANN is a novel single neuron based architecture first proposed by Pao **[11]**. It has been shown that this network may be conveniently used for functional approximation and pattern classification with faster convergence rate and lesser computational load than a Multi-layer Perceptron (MLP) structure.

The structure of the FLANN is fairly simple. It is a flat net without any need for a hidden layer. Therefore, the computations as well as learning algorithm used in this network are simple. The functional expansion of the input to the network effectively increases the dimensionality of the input vector and hence the hyper-planes generated by the FLANN provide greater discrimination capability in the input pattern space. Various system identifications, control of nonlinear systems, noise cancellation and image classification systems **[12]** have been reported in recent times. These experiments have proven the ability of FLANN to give out satisfactory results to problems with highly non-linear and dynamic data **[13]**. Further the ability of the FLANN architecture based model to predict stock index movements, both for short term (next day) and medium term (one month and two

months) prediction using statistical parameters consisting of well known technical indicators based on historical index data is shown and analyzed.

### 3.2 Structure of Functional Linked ANN

FLANN is a single layer, single neuron architecture, first proposed by Pao **[11]**, which has the exceptional capability to form complex decision regions by creating non-linear decision boundaries. The architecture of the FLANN is different from the linear weighting of the input pattern produced by the linear links of the better known Multi Layer Perceptron (MLP). In a FLANN, each input to the network undergoes functional expansion through a set of basis functions. The functional link acts on an element or the entire pattern itself by generating a set of linearly independent functions. The inputs expanded by a set of linearly independent functions in the function expansion block, causes an increase in the input vector dimensionality. This enables FLANN to solve complex classification problems by generating non-linear decision boundaries. In our experiment, the functional expansion block comprises of a set of trigonometric function.

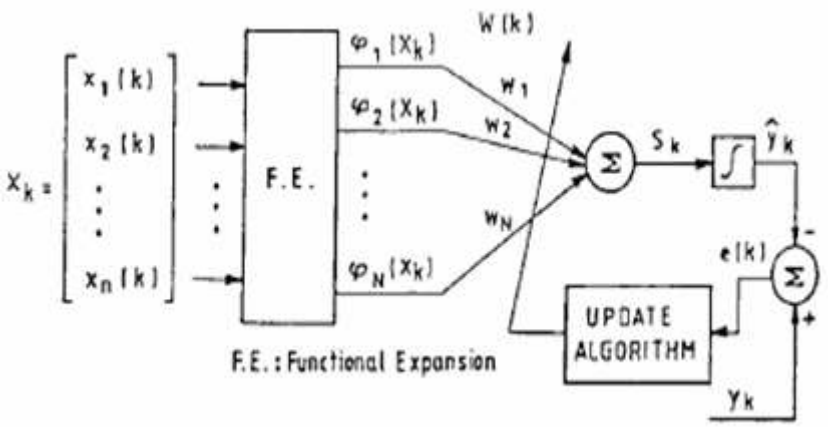



Figure 3.1 The figure shows structure of FLANN with single output

The basis functions for the FLANN, $B = \{\phi_i \in \mathrm{L}(\mathrm{A})\}_{i\in x}$ is to be selected keeping the following properties into consideration: **1)** $\phi_i = 1$, **2)** the subset $B_j = \{\phi_i \in B\}_{i=1}^{j}$ is a linearly independent set, i.e., if $\sum_{i=1}^{N} w_i\phi_i = 0$, then $w_i = 0$ for all $i = 1, 2, 3....j$ , and **3)** $\sup_j \left[\sum_{i=1}^{j} \| \phi_i \|_A^2\right]^{1/2} < \infty$. Let $B_N = \{\phi\}_{i=1}^{N}$ be a set of basis functions to be considered to the FLANN as shown in fig. 1. Thus, the FLANN consists of N basis functions $\{\phi_1, \phi_2, \phi_3, ....., \phi_N\} \in B_N$ with following input-output relationship for the *j*th output

$\hat{y}_j = \rho(S_j)$;

Where, $$S_j = \sum_{i=1}^{N} w_{ji}\phi_i(X) \qquad (1)$$

Where $X \in A \subset R^n$, i.e., $X = \left[x_1 x_2 \ldots x_n\right]^T$ is the input pattern vector, $\hat{y} \in R^m$, i.e., $\hat{y} = [\hat{y}_1 \hat{y}_2 \ldots \hat{y}_m]^T$ is the output vector and $w_j = [w_{j1} w_{j2} \ldots w_{jN}]$ is the weight vector associated with the **j** th output of the network. The non-linear function considered in this case $\rho(\bullet) = \tanh(\bullet)$.

Considering the m-dimensional input vector, (a) can be written as

$$S = W\Phi$$

Where W is (m×N) weight matrix of FLANN given by, $W = [w_1 w_2 ... w_m]^T$, $\phi = [\phi_1(X)\phi_2(X)...\phi_N(X)]^T$ is the basis function vector, and $S = [S_1 S_2 ... S_N]^T$ is a matrix of linear outputs of FLANN. The m-dimensional output vector $\hat{y}$ may be given by

$$\hat{y} = \rho(S) = f_W(X) \qquad (3)$$

### 3.3 Learning with Functional Linked ANN

The learning of ANN can be described as approximating a continuous, multivariate function f(X) by an approximating function $f_w(X)$ .Given a function the objective of the learning algorithm is to find the optimum weights such that $f_w(X)$ obtained approximates f(X) within an error e. This is achieved by recursively updating the weights. Let the training sequence be denoted by $\{X_k, y_k\}$ and the weight of the network be W(k), where k is the discrete time index given by k= κ +λ K where λ=0,1,2,….., and κ = 0,1,2,…,K. From (1) the j th output of FLANN at a given time k can be given as:

$$\hat{y}_j = \rho\left(\sum_{i=1}^{N} w_{ji}(k)\phi_i(X_k)\right)$$
$$= \rho(w_j(k)\phi^T(X_k)) \qquad (4)$$

For all $X \in A$ and j =1,2,3,….,m where $\phi = [\phi_1(X_k)\phi_2(X_k)...\phi_N(X_k)]$ .Let the corresponding error be denoted by $e_j(k) = y_j(k) - \hat{y}_j(k)$ .The Least Mean Square (LMS) update rule for all the weights of the FLANN is given by

$$W(k+1) = W(k) + \mu\delta(k)\phi(X_k) \qquad (5)$$

Where, $W = [w_1(k)w_2(k)...w_m(k)]^T$ is the M×N dimensional weight matrix of the FLANN at the k-th time instant is

$$\delta(k) = [\delta_1(k)\delta_2(k)...\delta_m(k)]^T,$$

$$\text{And } \delta_j(k) = (1 - \hat{y}_j(k)^2)e_j(k) \qquad (6)$$

Similarly the Recursive Least Square (RLS) update rule for all weights of the FLANN is given by

$$W(k+1) = W(k) + ej(k)zzk'(k) \quad (7)$$

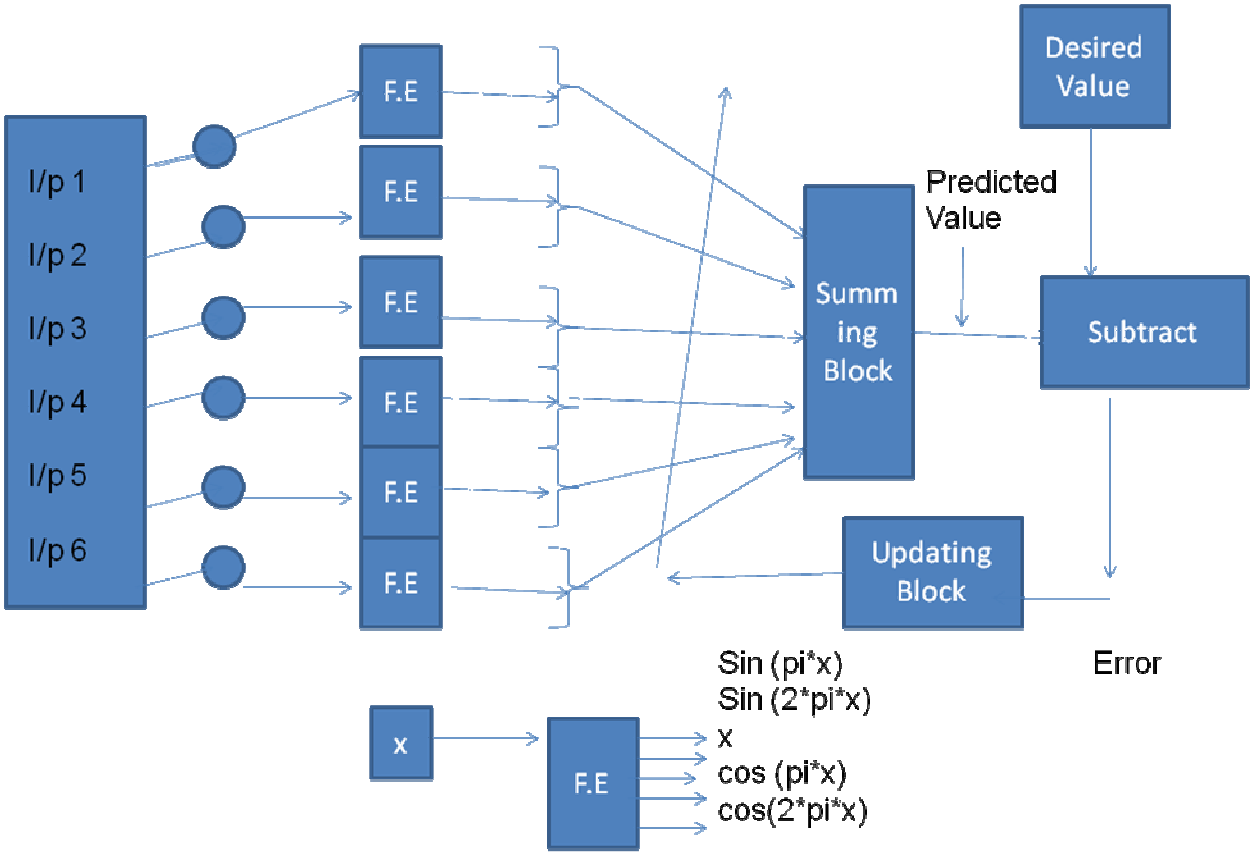
Where, zzk(k) = z(k) /(1+ q) , q = X (k).zk(k) and zk(k) = R(k).X (k) (8)

The autocorrelation matrix R(k) is updated with the equation,

$$R(k+1) = R(k) - zzk(k).zk(k)' \quad (9)$$

Which is initialized using the expression, $R(0) = \eta.I$ where I is the identity matrix and η is a constant.

The motivations for using trigonometric polynomials in the functional expansion stage are explained below. Of all the polynomials of N-th order with respect to an ortho-normal system $\{\varphi_i(x)\}_{i=1}^{N}$ gives the best approximation in the metric space $L^2$ is given by the N-th partial sum of its Fourier series with respect to this system. Thus, the trigonometric polynomial basis functions given by $\{1,\cos(\pi x),\sin(\pi x),\cos(2\pi x),\sin(2\pi x),....,\cos(N\pi x),\sin(N\pi x)\}$ provide a compact representation of the function in the mean square sense. However, when the outer product terms are used along with the trigonometric polynomials for function expansion, better results were obtained in the case of learning of a two-variable function.




**Learning with Functional Linked ANN**

### 3.4 Introduction to Radial Basis Function models

A radial basis function (RBF) is a real-valued function whose value depends only on the distance from the origin, so that $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$; or alternatively on the distance from some other point *c*, called a *center*, so that $\phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$. Any function φ that satisfies the property $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$ is a radial function. The norm is usually Euclidean distance, although other distance functions are also possible. For example by using probability metric it is for some radial functions possible[1] to avoid problems with ill conditioning of the matrix solved to determine coefficients $w_i$ (see below), since the $\|\mathbf{x}\|$ is always greater than zero. Sums of radial basis functions are typically used to approximate given functions. This approximation process can also be interpreted as a simple kind of neural network. The RBF Network is a multilayer feed forward network with a single layer hidden unit which operates as "Kernel" nodes. As such, it represents an alternative to the multilayer perceptrons. Advantages of RBF networks over multilayer perceptrons trained with the back –propagation algorithm include a more straightforward training process and a simple network structure. Ordinarily, development of RBF networks is pursued assuming real data and real free parameter. In particular, we are given a set of data points in the observation space defined by specified values of input signal and a desired response, and the requirement is to find an input-output mapping that passes through these points.

### 3.5 Radial Basis Function structure

RBF networks differ from multilayer networks in the following structural respects:

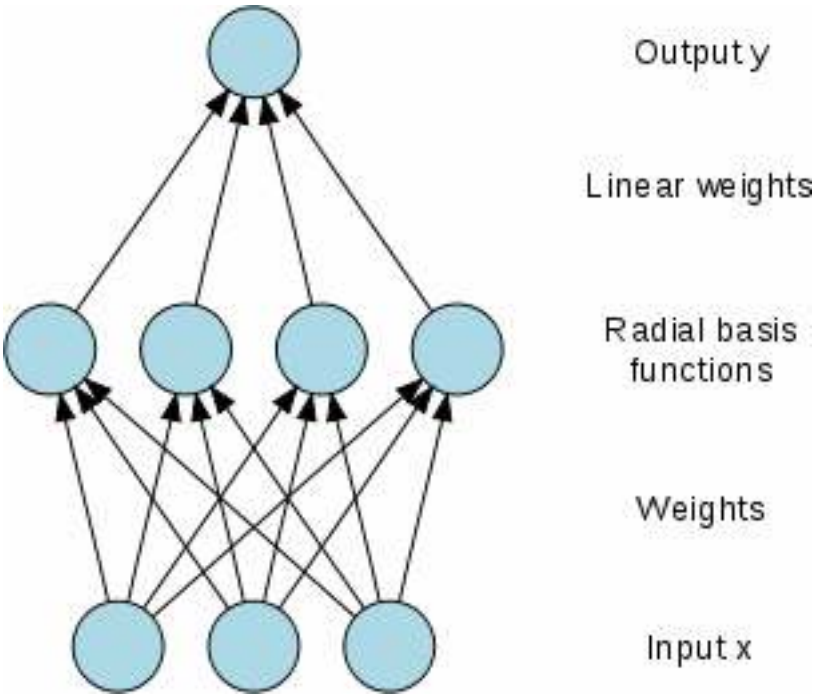
- RBF networks have a single hidden layer ,whereas multilayer perceptrons have one or more hidden layers.
- In RBF networks ,the transfer functions connecting the input layer to the hidden layer are nonlinear and those connecting the hidden layer to the output layer are linear.In multilayer perceptron, the transfer functions of each hidden layer connecting it to the previous layer are all nonlinear and the transfer functions of theoutput layer may be nonlinear or linear ,depending on application of interest.
- Each hidden layer of an RBF Network computes a distance function between the input vector and the center of a radial basis function characterizing that particular unit.Onthe other hand , each neuron of a multilayer perceptron computes the inner product of the input vector applied to that neuron and the vector of associated synaptic weights.

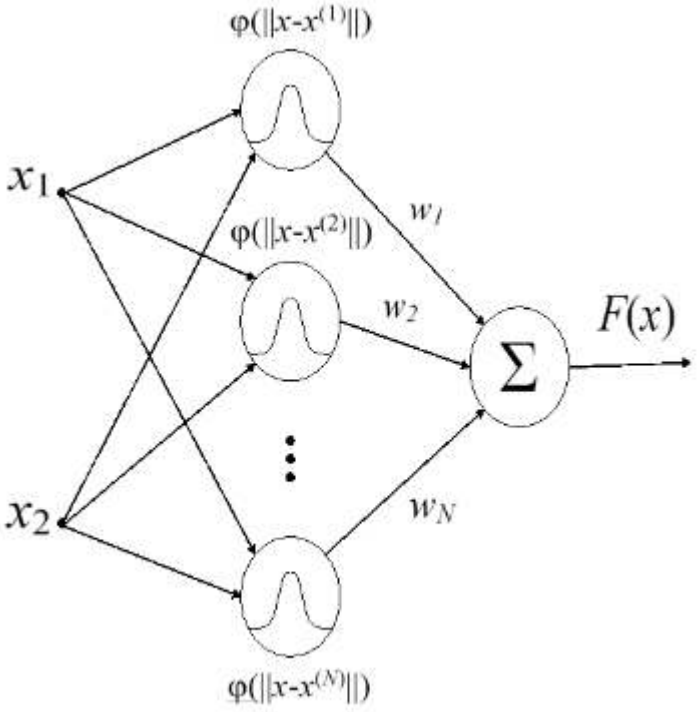
Gaussian Function is

$$\varphi(\mathbf{u}; \mathbf{t}_k) = \exp\left( -\frac{1}{\sigma_k^2} \|\mathbf{u} - \mathbf{t}_k\|^2 \right),$$

Where $t_k$ is the center $\sigma_k$ is the width and $\|u - t_k\|$ denotes the distance between u and center $t_k$. Now we cam formulate the input-output mapping realized by a Gaussian RBF function network as follows

$$y = \sum_{k=1}^{K} w_k \exp(-\frac{1}{\sigma_k^2} \|\mathbf{u} - \mathbf{t}_k\|^2)$$






### 3.6 Learning Methods

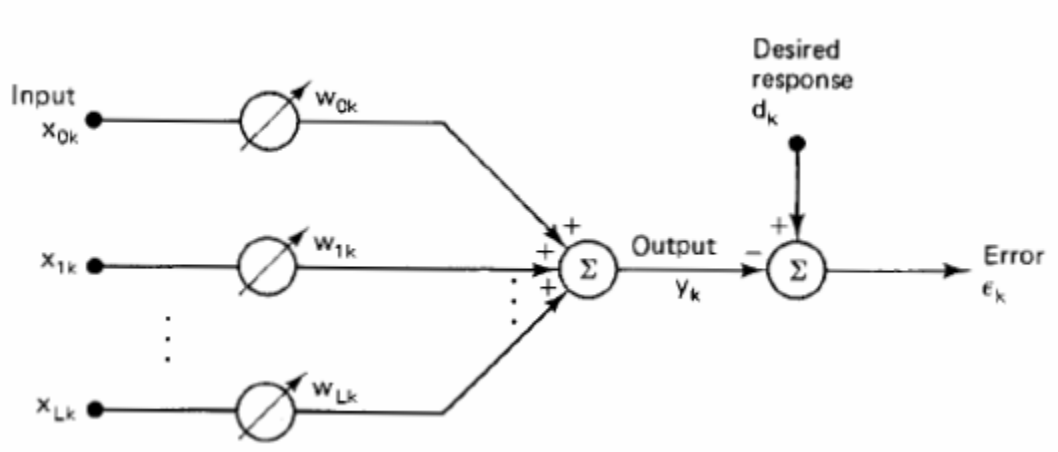
### 3.6.a Least Mean Square method

Least mean squares (LMS) algorithms are used in adaptive filters to find the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in which the filter is adaptive based on the error at the current time. It was invented in 1960 by Stanford University professor Bernard Widrow and his first Ph.D. student, Ted Hoff.

The adaptive linear combiner output $y_k$ is a linear combination of the input samples. The error in measurement is given by

$$\varepsilon_k = d_k - X_K^T W_K$$

where $X_K^T$ is the transpose vector of input samples.




To develop an adaptive algorithm ,it is required to estimate the gradient of ξ=E[$\varepsilon_k^2$] by taking differences between short term averages of $\varepsilon_k^2$.Instead, to develop the LMS algorithm process, $\varepsilon_k^2$is taken as the estimate of $\varepsilon_k$ Thus at each iteration in the adaptive process a gradient estimate form is as follows:

$$\hat{\nabla}_k = \begin{bmatrix} \frac{\partial \varepsilon_k^2}{\partial w_0} \\ \vdots \\ \frac{\partial \varepsilon_k^2}{\partial w_L} \end{bmatrix} = 2\varepsilon_k \begin{bmatrix} \frac{\partial \varepsilon_k}{\partial w_0} \\ \vdots \\ \frac{\partial \varepsilon_k}{\partial w_L} \end{bmatrix} = -2\varepsilon_k \mathbf{X}_k$$

With this simple estimate the steepest descent type of adaptive algorithm is specified as

$$W_{k+1} = W_k - \mu\nabla_k$$
$$= W_k + 2\mu\varepsilon_k X_k$$

Where "μ" is the **gain constant** that regulates the speed and stability of adaptation. Since the weight changes at every iteration are based on imperfect gradient estimates, the adaptive process is expected to be noisy. The LMS algorithm can be implemented without squaring, averaging or differentiation and is simple and efficient process.

As with all adaptive Algorithms, the primary concern with the LMS Algorithm is its convergence to the weight vector solution, where error E [$\varepsilon_k^2$] is minimized.

LMS algorithm used to update parameters of RBF strucure:

The RBF structure has the following parameters

1 The n centres of RBF where there are n nodes in the hidden layer

2 The n variance of RBF where there are n nodes in hidden layer

3 The weights in the output layer

The network parameters of RBF are updated using LMS by the following equations:-

$$y(n) = \sum_{k=1}^{K} w_k(n)\phi(\mathbf{u}(n); \mathbf{t}_k(n))$$

$$e(n) = d(n) - y(n)$$

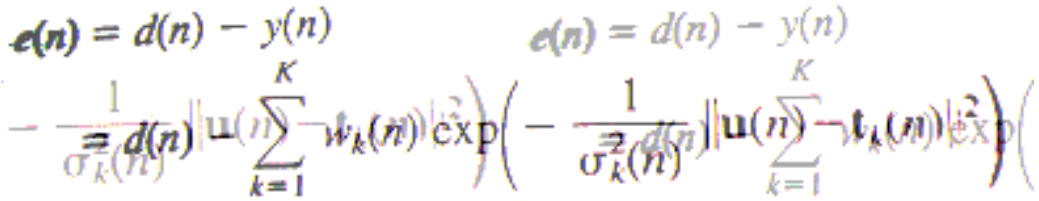
$$\mathbf{w_k}(n+1) = w_k(n) + \mu_w e^*(n)\, \varphi(\mathbf{u}(n); \mathbf{t}_k(n))$$

$$\mathbf{t_k}(n+1) = \mathbf{t_k}(n) + 2\mu_t e^*(n)\, w_k(n)\varphi(\mathbf{u}(n); \mathbf{t}_k(n))\, \frac{\mathbf{u}(n) - \mathbf{t_k}(n)}{\sigma_k^2(n)}$$

$$\sigma_k^2(n+1) = \sigma_k^2(n) + \mu_\sigma\, e^*(n)\, w_k(n)\varphi(\mathbf{u}(n); \mathbf{t}_k(n))\, \frac{\|\mathbf{u}(n) - \mathbf{t_k}(n)\|}{\sigma_k^2(n)}$$

where e(n) is the error signal produced in response to the nth example,and N is the total number of examples in the training set.

Error signal is defined by:



### 3.6.b Particle Swarm Optimization (PSO)

Particle swarm optimization is a stochastic, population-based computer algorithm for problem solving. It is a kind of swarm intelligence that is based on social-psychological principles and provides insights into social behavior, as well as contributing to engineering applications. The particle swarm optimization algorithm was first described in 1995 by James Kennedy and Russell C. Eberhart. The techniques have evolved greatly since then, and the original version of the algorithm is barely recognizable in the current ones.

Social influence and social learning enable a person to maintain cognitive consistency. People solve problems by talking with other people about them, and as they interact their beliefs, attitudes, and behaviors change; the changes could typically be depicted as the individuals moving toward one another in a socio-cognitive space.

The particle swarm simulates this kind of social optimization. A problem is given, and some way to evaluate a proposed solution to it exists in the form of a fitness

function. A communication structure or social network is also defined, assigning neighbors for each individual to interact with. Then a population of individuals defined as random guesses at the problem solutions is initialized. These individuals are candidate solutions. They are also known as the particles, hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion. The particles iteratively evaluate the fitness of the candidate solutions and remember the location where they had their best success. The individual's best solution is called the particle best or the local best. Each particle makes this information available to their neighbors. They are also able to see where their neighbors have had success. Movements through the search space are guided by these successes, with the population usually converging, by the end of a trial, on a problem solution better than that of non-swarm approach using the same methods.

The swarm is typically modeled by particles in multidimensional space that have a position and a velocity. These particles fly through hyperspace (i.e., $\mathbb{R}^n$) and have two essential reasoning capabilities: their memory of their own best position and knowledge of the global or their neighborhood's best. In a minimization optimization problem, problems are formulated so that "best" simply means the position with the smallest objective value. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. So a particle has the following information to make a suitable change in its position and velocity:

- A global best that is known to all and immediately updated when a new best position is found by any particle in the swarm.
- Neighborhood best that the particle obtains by communicating with a subset of the swarm.
- The local best, which is the best solution that the particle has seen

The particle position and velocity update equations in the simplest form that govern the PSO are given by:

$v(t+1) = v(t) + c1*rand*(pbest - x(t))+c2*rand\ (gbest - x(t))$

$x(t+1) = x(t) + v(t+1)$

As the swarm iterates, the fitness of the global best solution improves (decreases for minimization problem). It could happen that all particles being influenced by the global best eventually approach the global best, and from there on the fitness never improves despite however many runs the PSO is iterated thereafter. The particles also move about in the search space in close proximity to the global best and not exploring the rest of search space. This phenomenon is called 'convergence'. If the inertial coefficient of the velocity is small, all particles could slow down until they approach zero velocity at the global best. The selection of coefficients in the velocity update equations affects the convergence and the ability of the swarm to find the optimum. One way to come out of the situation is to reinitialize the particles positions at intervals or when convergence is detected.

Some research approaches investigated the application of constriction coefficients and inertia weights. There are numerous techniques for preventing premature convergence. Many variations on the social network topology, parameter-free, fully adaptive swarms, and some highly simplified models have been created. The algorithm has been analyzed as a dynamical system, and has been used in hundreds of engineering applications; it is used to compose music, to model markets and organizations, and in art installations.

A single particle by itself is unable to accomplish anything. The power is in interactive collaboration.

Let $f : \mathbb{R}^m \to \mathbb{R}$ be the fitness function that takes a particle's solution with several components in higher dimensional space and maps it to a single dimension metric. Let there be *n* particles, each with associated positions $\mathbf{x}_i \in \mathbb{R}^m$ and velocities $\mathbf{v}_i \in \mathbb{R}^m$, $i = 1, \ldots, n$. Let $\hat{\mathbf{x}}_i$ be the current best position of each particle and let $\hat{\mathbf{g}}$ be the global best.

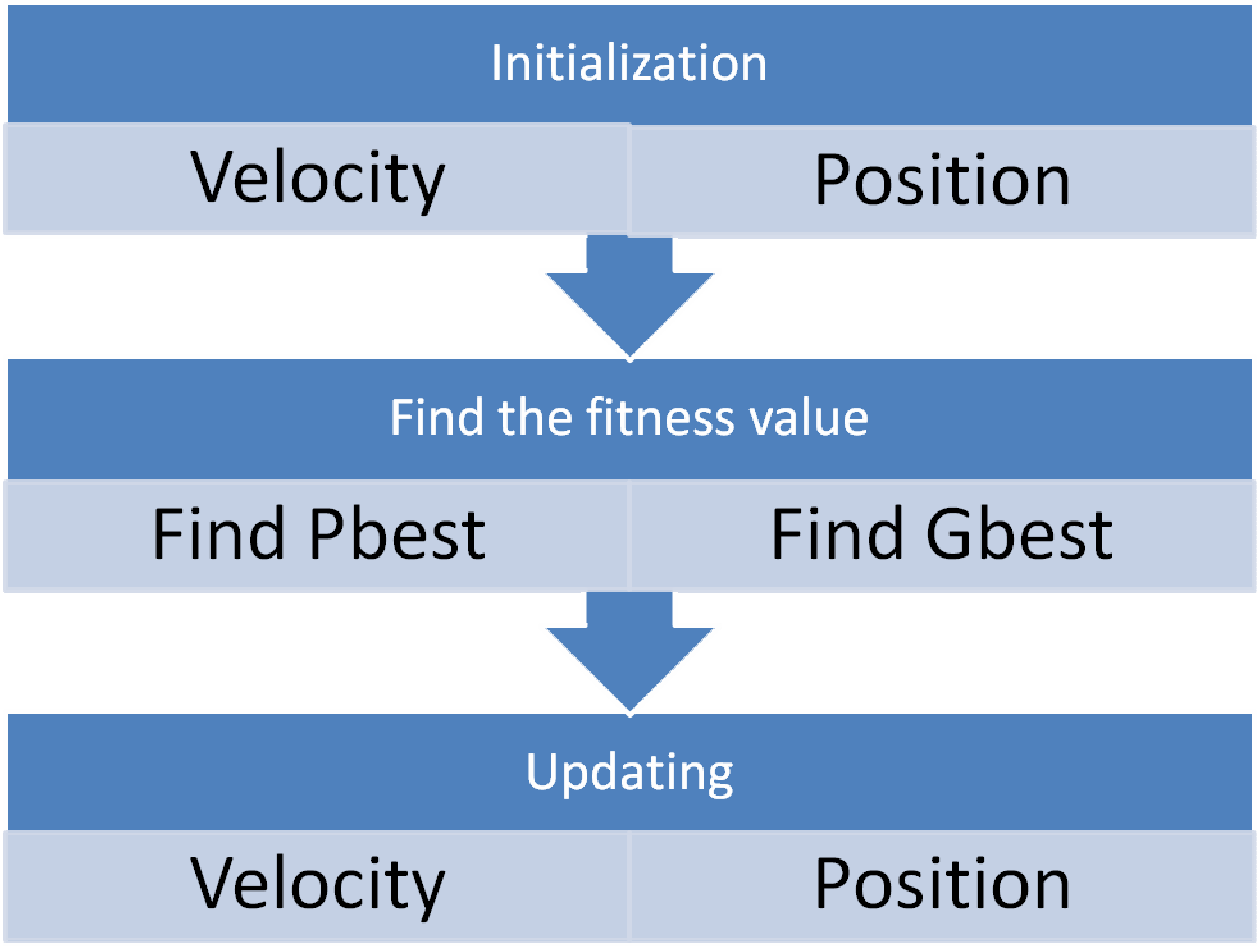
- Initialize $\mathbf{x}_i$ and $\mathbf{v}_i$ for all *i*. One common choice is to take $\mathbf{x}_{ij} \in U[a_j, b_j]$ and $\mathbf{v}_i = \mathbf{0}$ for all *i* and $j = 1, \ldots, m$, where $a_j$,$b_j$ are the limits of the search domain in each dimension, and *U* represents the uniform distribution (continuous).
- $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i$ and $\hat{\mathbf{g}} \leftarrow \arg\min_{\mathbf{x}_i} f(\mathbf{x}_i), i = 1, \ldots, n$.
- While not converged:
  - For each particle $1 \leq i \leq n$:
    - Create random vectors $\mathbf{r}_1$, $\mathbf{r}_2$: $\mathbf{r}_{1j}$ and $\mathbf{r}_{2j}$ for all *j*,by taking $\mathbf{r}_{1j}, \mathbf{r}_{2j} \in U[0, 1]$ for $j = 1, \ldots, m$
    - Update the particle positions: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$.
    - Update the particle velocities: $\mathbf{v}_i \leftarrow \omega \mathbf{v}_i + c_1 \mathbf{r}_1 \circ (\hat{\mathbf{x}}_i - \mathbf{x}_i) + c_2 \mathbf{r}_2 \circ (\hat{\mathbf{g}} - \mathbf{x}_i)$.
    - Update the local bests: If $f(\mathbf{x}_i) < f(\hat{\mathbf{x}}_i)$, $\hat{\mathbf{x}}_i \leftarrow \mathbf{x}_i$.
    - Update the global best If $f(\mathbf{x}_i) < f(\hat{\mathbf{g}})$, $\hat{\mathbf{g}} \leftarrow \mathbf{x}_i$.
- $\hat{\mathbf{g}}$ is the optimal solution with fitness $f(\hat{\mathbf{g}})$.

Note the following about the above algorithm:

- ω is an inertial constant. Good values are usually slightly less than 1.

- $c_1$ and $c_2$ are constants that say how much the particle is directed towards good positions. They represent a "cognitive" and a "social" component, respectively, in that they affect how much the particle's personal best and the global best (respectively) influence its movement. Usually we take $c_1, c_2 \approx 2$.
- $\mathbf{r}_1, \mathbf{r}_2$ are two random vectors with each component generally a uniform random number between 0 and 1.
- $\circ$ operator indicates element-by-element multiplication.

When PSO is incorporated to update parameters of RBF structure, the different parameters of RBF including the mean, variance and weights of the output layer are to be updated. These parameters as a whole represent a particle and each particle searches for the solution by checking the fitness function. Each particle gets its own best position called pbest in every iterations. Out of these pbest the best fit solution gives the global best value called gbest.




**Flowchart depicting a general PSO algorithm**

# Chapter 4

# NETWORK INPUT SELECTION DATA PREPROCESSING

The data for the stock market prediction experiment has been collected for three different stock indices namely Dow Jones Industrial Average (DJIA), USA, Standards & Poor's 500 Index (S&P 500), USA and Bombay Stock Exchange (BSE). The time series data of all the stock indices were collected from 3rd January 1994 to 23rd October 2006. Thus there were 3228 data patterns for DJIA and 2000 data patterns for S&P 500 index. The data collected for the stock indices consisted of the closing price, opening price, and lowest value in the day, highest value in the day and the total volume of stocks traded in each day. (Note that one day's closing price of the index can be slightly different from next day's opening price, due to introduction of afterhours trading between institutions private exchanges). The proposed forecasting model is developed to forecast the closing price of the index in each day of the forecasting period.

Different technical and fundamental indicators are used as inputs to the network. Technical indicators are any class of metrics whose value is derived from generic price activity in a stock or asset. Technical indicators look to predict the future price levels, or simply the general price direction, of a security by looking at past patterns. Out of the many technical indicators used by traders, 10 indicators have been chosen as input to the network which has been used before by many researchers for stock market forecasting problems. The details of the parameters and how they are calculated from the available data is given below:

• **Simple Moving Average (SMA):**

It's the simple average of the values by taking a window of the specified period. The various SMAs used in the experiment are:

1. 10 days (SMA10)
2. 20 days (SMA20)
3. 30 days (SMA30)

• **Accumulation/Distribution Line (ADO):**

It measures money flow in the security. It attempts to measure the ratio of buying to selling by comparing price movements of a period to the volume of that period.

ADO = ((Close – Low) – (High – Close))/ (High – Low) * Period's Volume

Every day's ADO has been taken in the experiment.

• **On Balance Volume (OBV):**

It is a momentum indicator that relates volume to price change.

Calculation of OBV:

If today's close > Yesterday's Close

OBV = Yesterday's OBV + Today's Volume

If today's close > Yesterday's Close

OBV= Yesterday's OBV – Today's volume

• **Williams %R (WILLIAMS):**

It is a momentum indicator that measures overbought/oversold levels.

Calculation of Williams %R =

(Highest high in n periods – Today's close)*100

(Highest high in n-periods – Lowest low in n-periods)

For this experiment: n= 9 days

• **Price Rate of Change (PROC):**

The PROC indicator displays the difference between the current price and closing price x-time periods ago.

Calculation:

(Today's close – Close x-periods ago) * 100

(Close x-periods ago)

Through experimental results it's found that x=10 are considered best for technical analysis

We have chosen the above five parameters so that our analysis takes into account various factors into consideration including momentum factor, volume of trade and any random spurt in prices which are generally unstable. These five parameters along with the present day input form the six inputs into the system.

However, many other parameters can be derived from the input data which portray various behaviors of the data set .These parameters have been systematically tabulated in the table.

| Technical Indicators | Formula |
|---|---|
| Simple Moving Average (SMA) | $\frac{1}{N}\sum_{i=1}^{N} x_i$<br>N = No. of Days. $x_i$ = today's price |
| Exponential Moving Average (EMA) | $(P \times A) + (\text{Previous EMA} \times (1 - A))$ ; A=2/(N+1)<br>P – Current Price, A- Smoothing factor, N-Time Period |
| Accumulation/ Distribution Oscillator (ADO) | $\frac{(\text{C.P - L.P}) - (\text{H.P - C.P})}{(\text{H.P - L.P}) \times (\text{Period's Volume})}$<br>C.P – Closing Price, H.P – Highest price, L.P – Lowest price |
| Stochastic Indicator (STOC) | $\%K = \frac{(\text{Today's Close - Lowest Low in K period})}{(\text{Highest High in K period - Lowest Low in K period})} \times 100$<br>$\%D$ = SMA of $\%K$ for the Period. |
| On Balance Volume (OBV) | If Today's Close > Yesterday's Close<br>OBV = Yesterday's OBV + Today's Volume<br>If Today's Close < Yesterday's Close<br>OBV = Yesterday's OBV – Today's Volume |

| | |
|---|---|
| WILLIAM's %R | $\%R = \frac{\text{(Highest High in n period - Today's Close)}}{\text{(Highest High in n period - Lowest Low in n period)}} \times 100$ |
| Relative Strength Index (RSI) | $\text{RSI} = 100 - \frac{100}{1 + (U/D)}$ |
| Price Rate Of Change (PROC) | $\frac{\text{(Today's Close - Close X-period ago)}}{\text{(Close X-period ago)}} \times 100$ |
| Closing Price Acceleration (CPAcc.) | $\frac{\text{( Close Price - Close Price N-period ago)}}{\text{(Close Price N-period ago)}} \times 100$ |
| High Price Acceleration (HPAcc.) | $\frac{\text{( High Price - High Price N-period ago)}}{\text{(High Price N-period ago)}} \times 100$ |

**Technical indicators and calculation**

# Chapter 5

# THE STOCK MARKET PREDICTION EXPERIMENT

Experiment Model Setup
Training Process
Testing Process

## 5.1 Experiment Model Setup

The data samples are taken from three stock exchange data .Data samples are collected from the historical values of Dow Jones's Industrial Average, Standard & Poor's and Bombay Stock Exchange data.

We employ models using Radial Basis Function and Functional Link Neural Network architecture (FLANN) structure where the parameters of each of the structure is updated using either LMS algorithm or PSO learning.

### 5.1.1 Modeling of FLANN Structure

FLANN is single neuron architecture. Each input is split up into five branches each being a distinct function of the primary input. Thus effectively we now have five times the primary inputs we had considered that go as inputs to the single neuron. For our experiment we have taken 6 input parameters for each pattern. For a 6 different statistical parameters of the stock index lag values, the total input to the single neuron FL-ANN is 30 plus a bias. This gives us 31 weights that are to be trained using a suitable adaptive algorithm for a particular stock index. The neuron adds up the input weight products and bias. The sum is then taken up by a suitable activation function to give the output of the network. For this particular case we used the tan hyperbolic activation function .The five distinct function applied to the each of branched input can be chosen as trigonometric functions, exponential functions, Chebychev polynomial functions. In the FLANN model of stock market prediction, four trigonometric functions namely Cos $\pi x$, Cos $2\pi x$, Sin $\pi x$ and Sin $2\pi x$ were used along with the variable $x$ itself. An optimum value of the convergence coefficient was taken as 0.1 for all the prediction experiments. The

inputs have to be normalized for the proper behavior of the network. The inputs are normalized to values between +1 and -1. This can be done by a number of normalization techniques. One of the popular techniques we used was expressing the data in terms of the maximum and minimum of the data set. All the values are normalized by using the following equation

$Y = (2*X – (Max + Min))/ (Max + Min)$

Where Y: - normalized values.

X: - present value.

### 5.1.2 Modeling of RBF structure

Radial Basis Function has one layer hidden layer having 6 centres.the activation function is a gaussian one which depends on the radial distance which is distance of input sample from the designated centres.Thus the first layer is a non linear dependancy.The resultant is multiplied by a weight corresponding to each centre and all of these are summed up to give a value which is called the plant output.The Radial Basis Function thus; have 6 centres (6 X 6 = 36 weights, as 6 input parameters are being fed to the RBF network), 6 variances and 6 linear weights corresponding to the 6 centres for each input sample.These parameters are trained using either least mean sqaure algorithm or particle swarm optimization.

The total data set of a particular stock market index is split up into two, one for training of the network and the rest for testing the performance of the network after freezing the weights. In this experiment we take approx 1000 to 2000 daily statistical data of the stock index as training set. The rest 600 values are set aside for testing. Out of 600 only 100 are shown in simulation graphs for clarity.

## 5.2 Training Process

The training of the network takes place in the following fashion. The weight update is epoch based. Since we have two different networks namely RBF and FLANN we will discuss the training process of each of them one by one.

As mentioned in section 5.1, altogether 31 weights of FLANN model need to be updated by learning algorithm. In PSO, these 31 weights constitute as a solution to the system. Each particle in solution space will have a 31 dimensional vector solution. In our simulation we have taken a swarm of 20 particles. Initially all particles are assigned random position. After the training, as described in section 3.6, all particles will converge to a gbest position which is a 31 dimensional vector. This vector is the optimal weight vector of FLANN model.

In RBF model, 6 X 6 = 36 weights (associated with mean), 6 for variance and 6 as linear weight constitute a solution to the model. All these weights are updated via LMS and PSO as described in section 3.6

The input data set are also normalized prior to the network training. The weights remain unchanged till all of the training data set is fed into the network, compared with the desired output and their respective error stored. The mean error for the entire epoch is calculated, and then the adaptive weight update takes place. The Least Mean Square (LMS) update algorithm and PSO is used in our experiment updates the weights by adding the product of the convergence constant, the respective input with the mean error for the epoch to the weights of the previous

epoch. The cost function for the training process is the Mean Square Error (MSE). It is suitable to end the training of the network when the minimum level of the cost function is observed. Thus for each iteration (epoch), the mean square error is calculated and plotted. Each of the iterations involves training the network with the 2500-odd patterns, calculation of mean error, weight update and representing the MSE. The number of iteration is decided upon by gradient of the MSE curve. If it is observed that there is no significant decrease in the MSE then the training experiment can be stopped. There exists a trade-off between the time taken and quality of training. High number of iterations tends to give better training of the network at the cost of time taken to train.

PSO is used to train parameters of structure using particles. We have used 20 particles whereby each particle represents a solution to the problem .These parameters as a whole represent one particle and each particle searches best solution to the problem. Each particle has a fitness value associated with it and it is error in our experiment .The aim is to make each particle search for the best fit solution and in turn minimize error. In this process each particle learns from its past solutions and determines the *pbest*- the best position for each particle in a particular iteration. In turn the particle learns from its neighbor and finds the best solution among the *pbest* and lands up with the global best solution called the *gbest*. The *pbest* and *gbest* help to update the various parameters.

### 5.3 Testing Process

At the end of the training process of the network, the weights are frozen for testing the network on inputs that were set apart from the training set. The testing set patterns are the input to the network and the output, the predicted index close price is compared with desired output or actual close price. The percentage of error is recorded for each data set. The criteria for judging the quality of prediction shown by the model is the mean of all the percentage error of the testing data set. . The Mean Absolute Percentage Error (MAPE) is used to gauge the performance of the trained prediction model for the test data. It is quite different from normal MSE, as evident from the equation below. In our simulation, we have calculated both MSE and MAPE, but the analysis and comparison is done on the basis of MAPE only. The effort is to minimize the MAPE for testing patterns in the quest for finding a better model for forecasting stock index price movements The MAPE is given as

$$MAPE = \frac{1}{N}\sum_{j=1}^{N}\left|\frac{y_j - \hat{y}_j}{y_j}\right| \times 100$$

# Chapter 6

# SIMULATION RESULTS

## 6.1 Simulation Results for DJIA

Out of around 3000 days data of DJIA, it was found that only 1000 days data is sufficent enough to train the various models for 1 day, 5 day, and 10 days ahead prediction.

For 30 and 60 day ahead prediction, upto 2000/2500 days data is used to train the network.

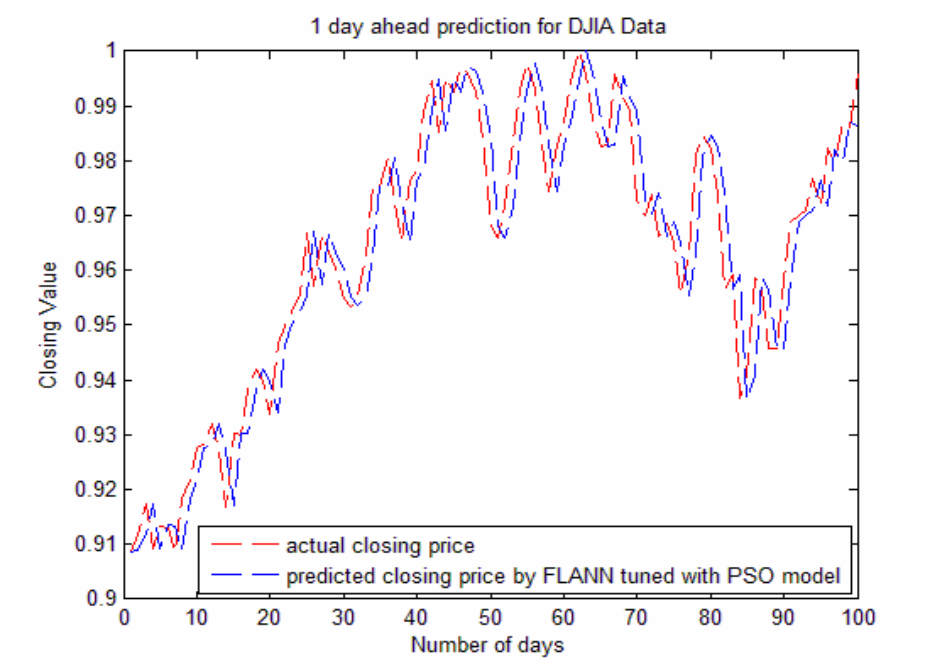
Model is tested with fresh 600 days data , out of which only 100 are shown for clarity.

Tuning Parameters and Associated Parameters:

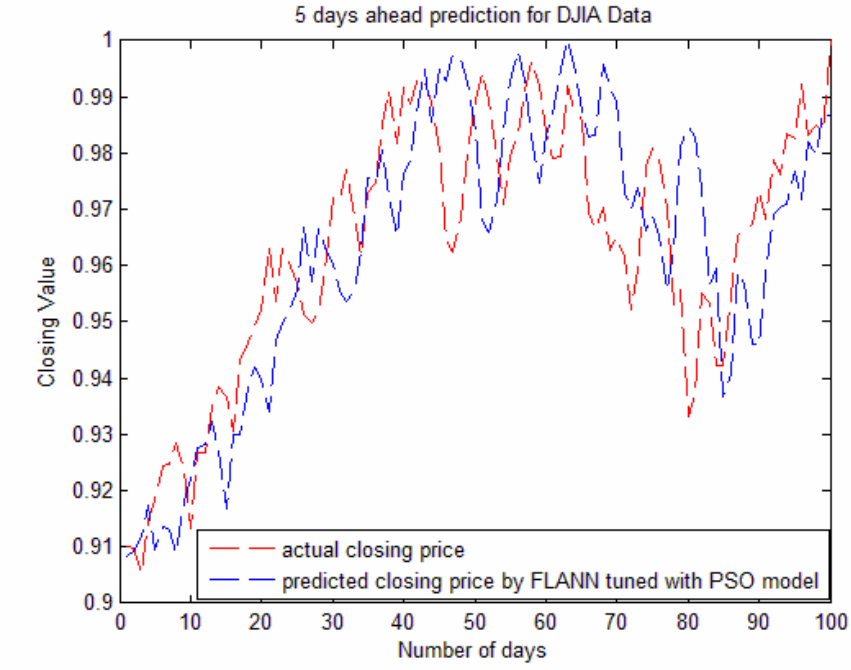
Particles = 20
μ(weight) = 0.4
μ(mean) = 0.5
μ(sig) = 0.7

### 6.1.1 Testing of FLANN Parameters tuned with PSO model

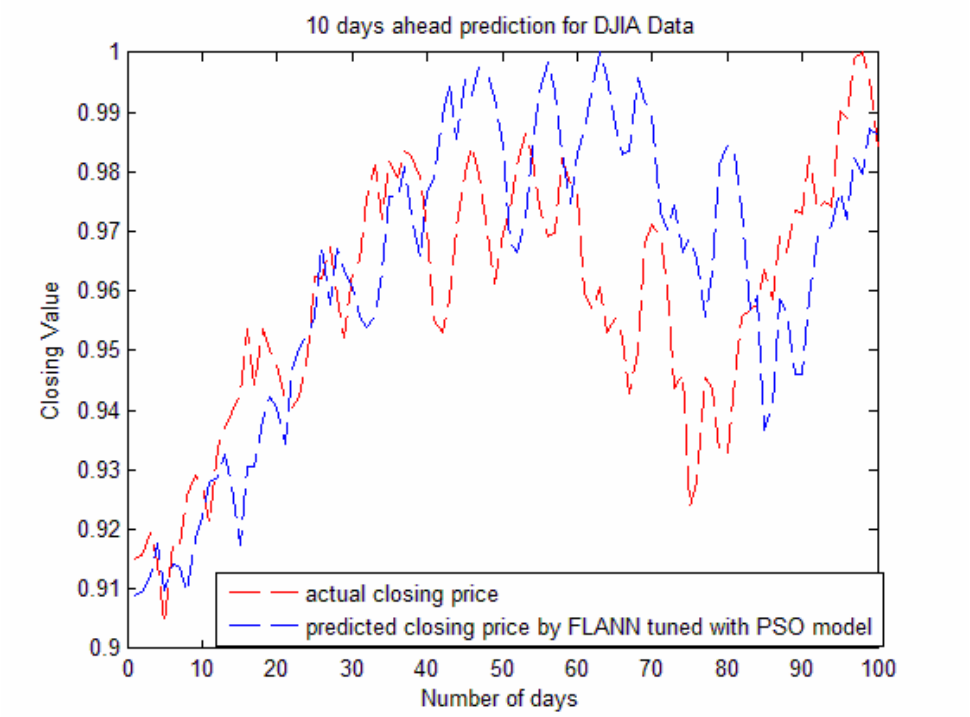
## For 1 day ahead prediction



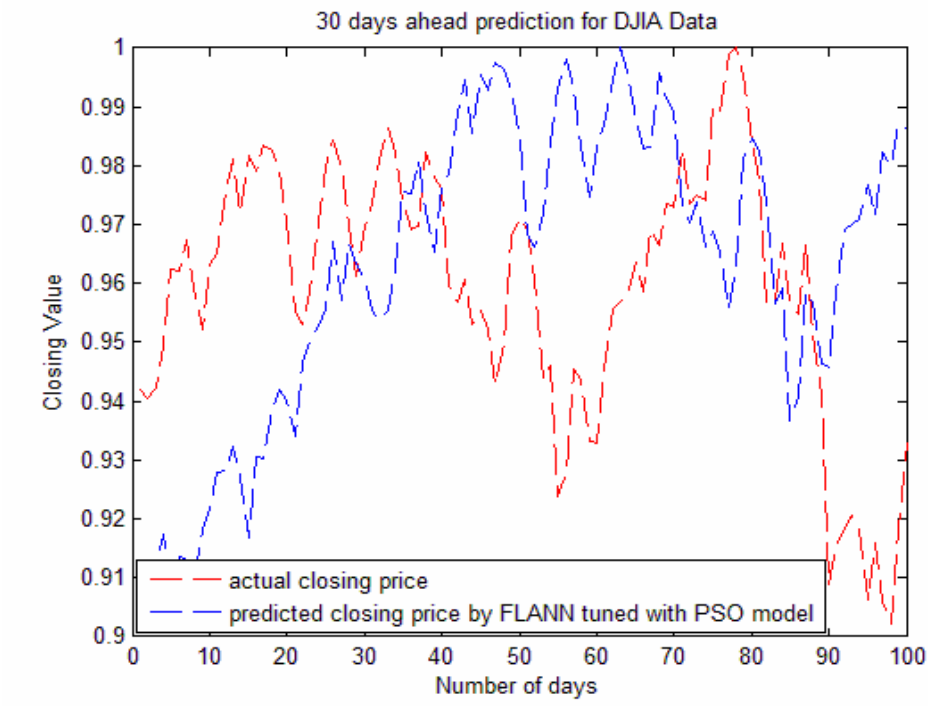

## For 5 days ahead prediction



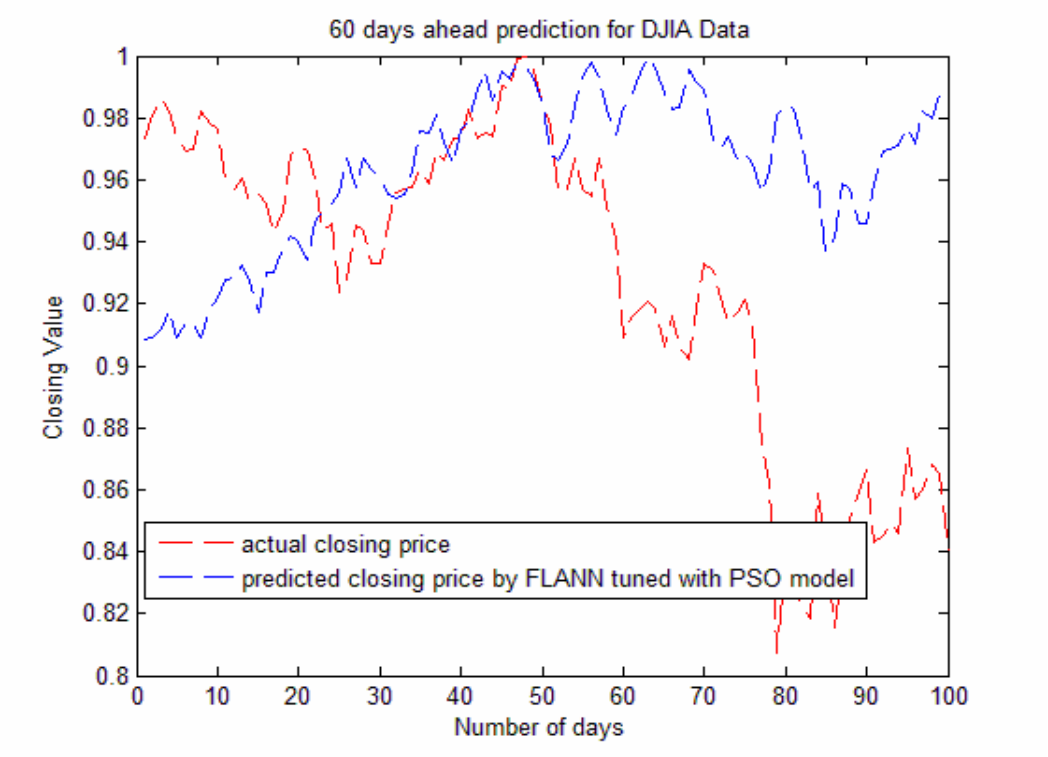
# For 10 days ahead prediction




# For 30 days ahead prediction

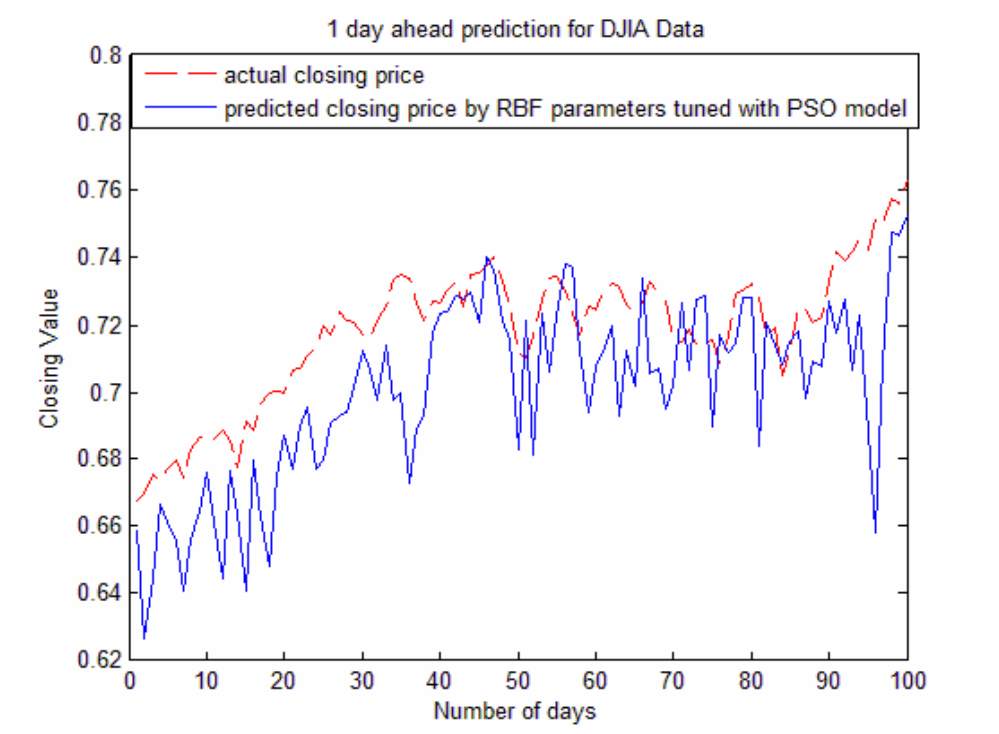


# For 60 days ahead prediction

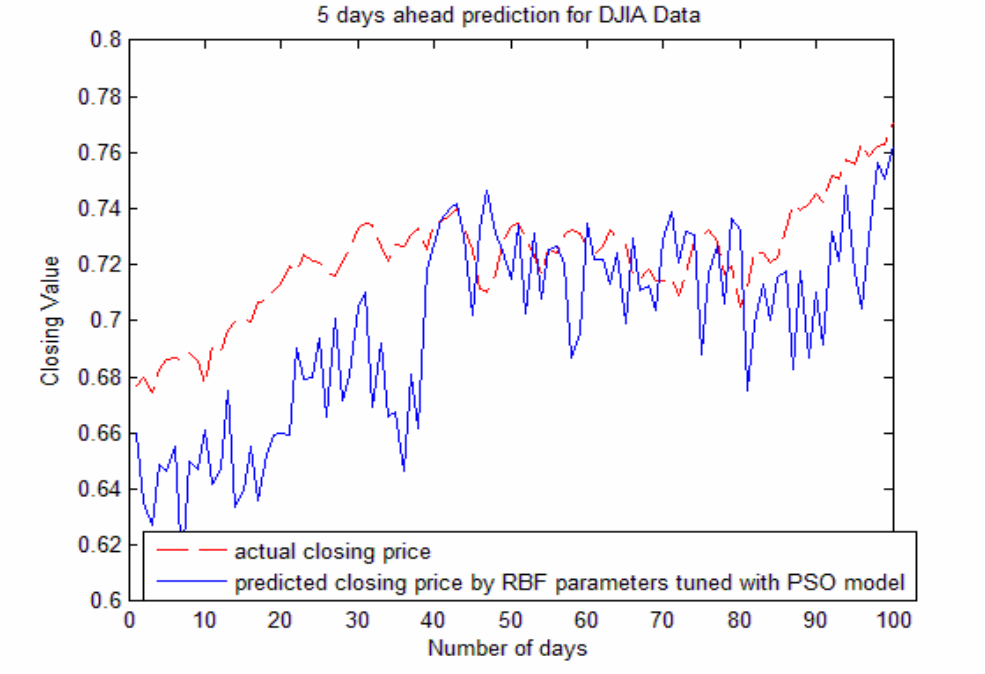


### 6.1.2 Testing of RBF Parameters tuned with PSO model

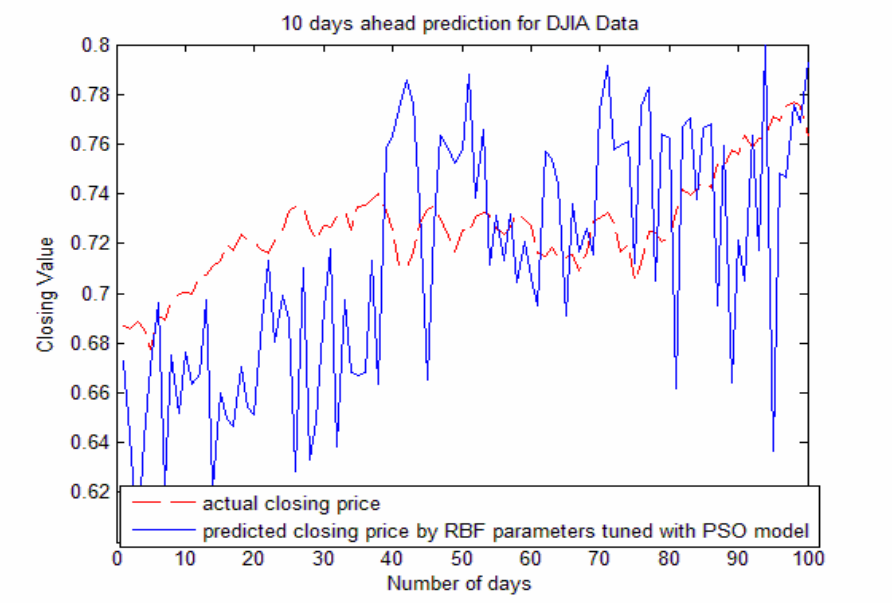
## For 1 day ahead prediction



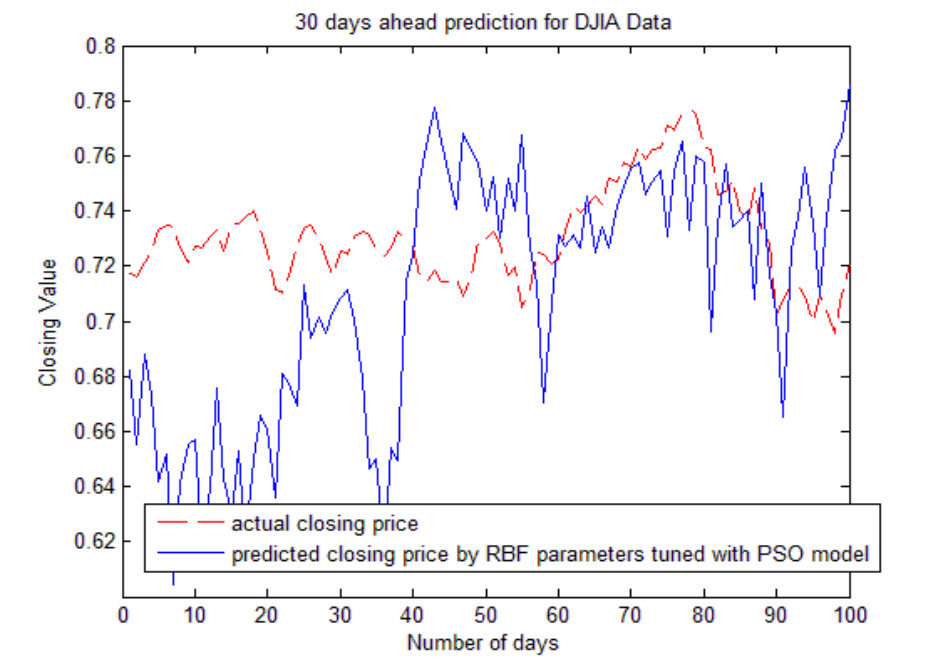

## For 5 days ahead prediction



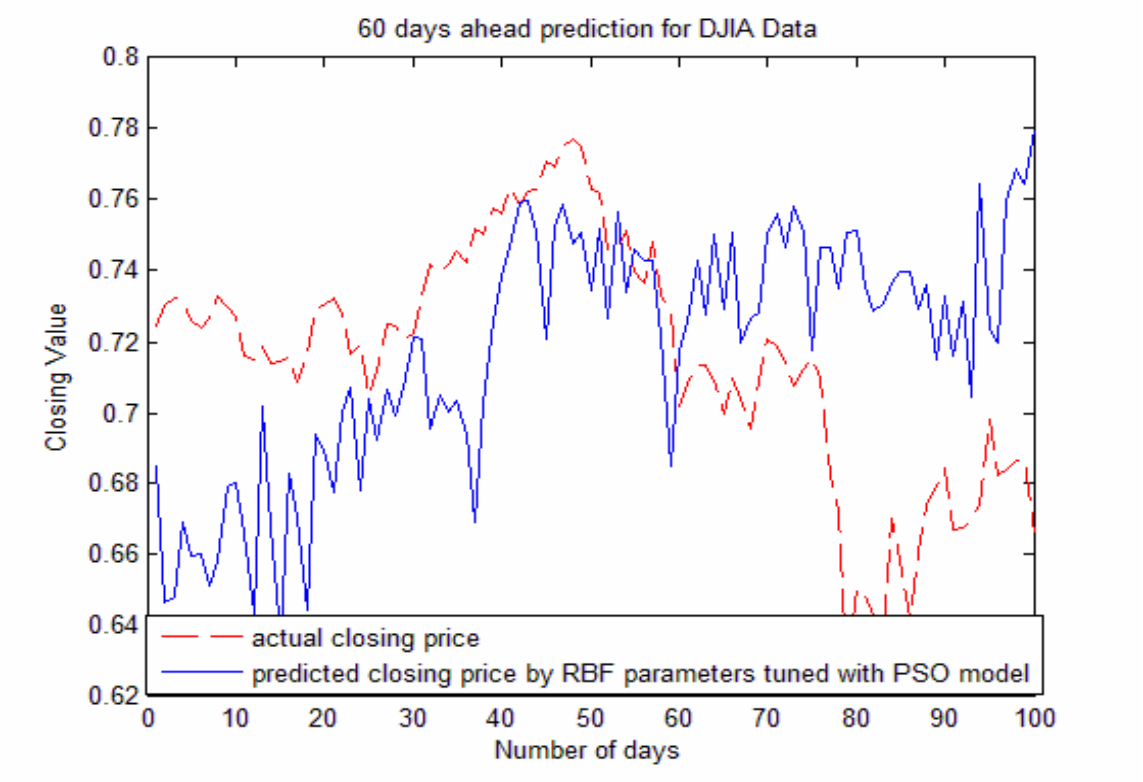
# For 10 days ahead prediction




# For 30 days ahead prediction

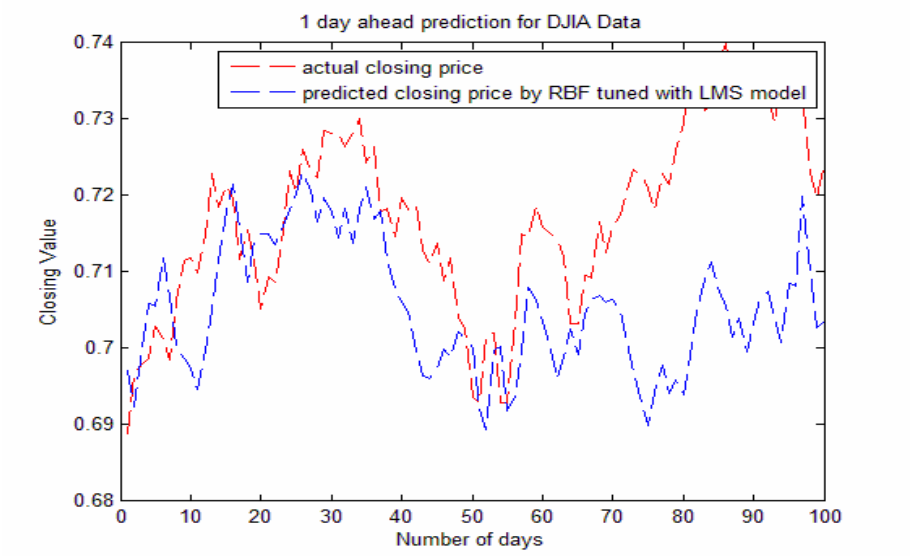


# For 60 days ahead prediction

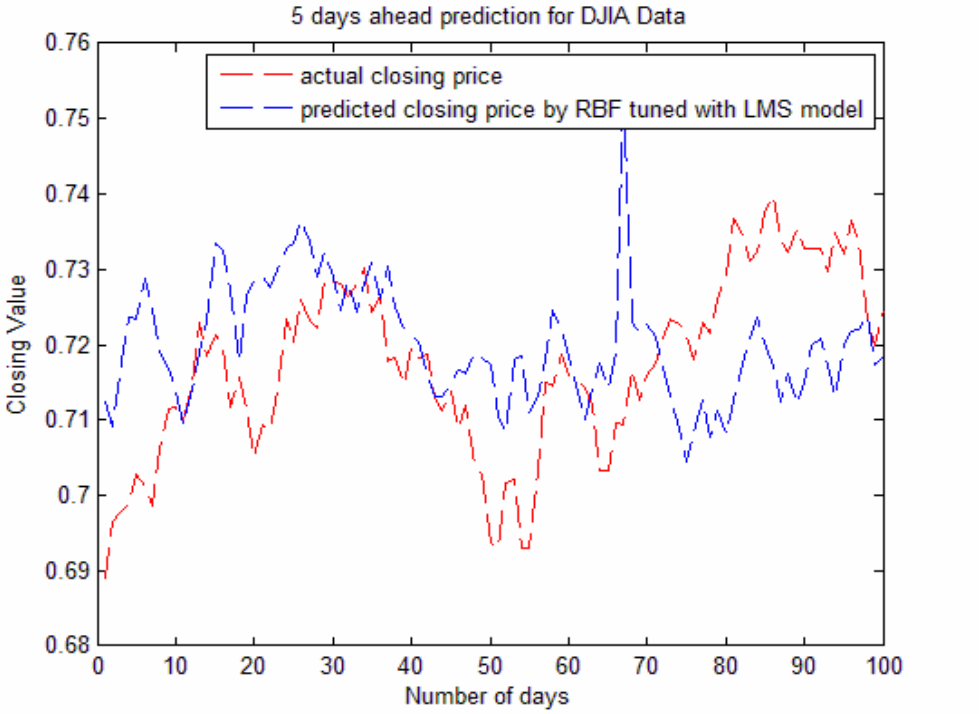


### 6.1.3 Testing of RBF Parameters tuned with LMS model

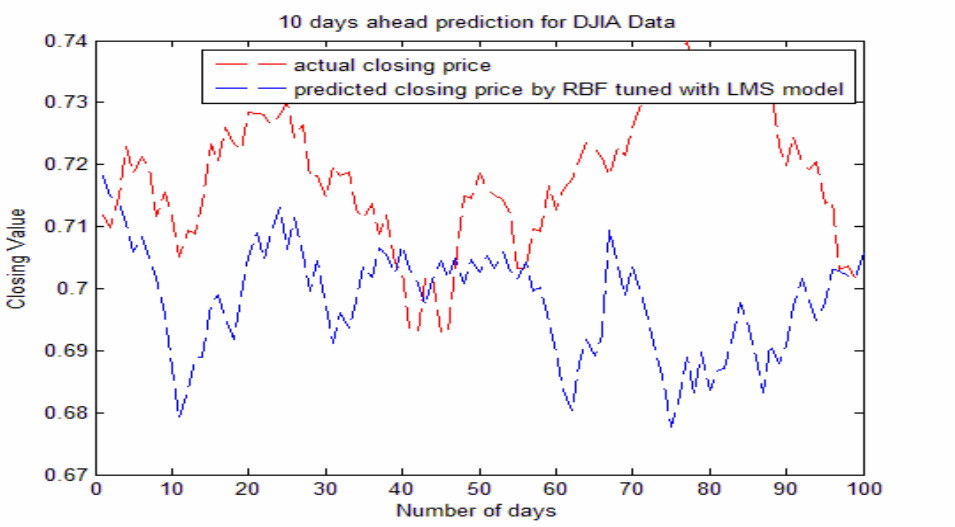
## For 1 day ahead prediction



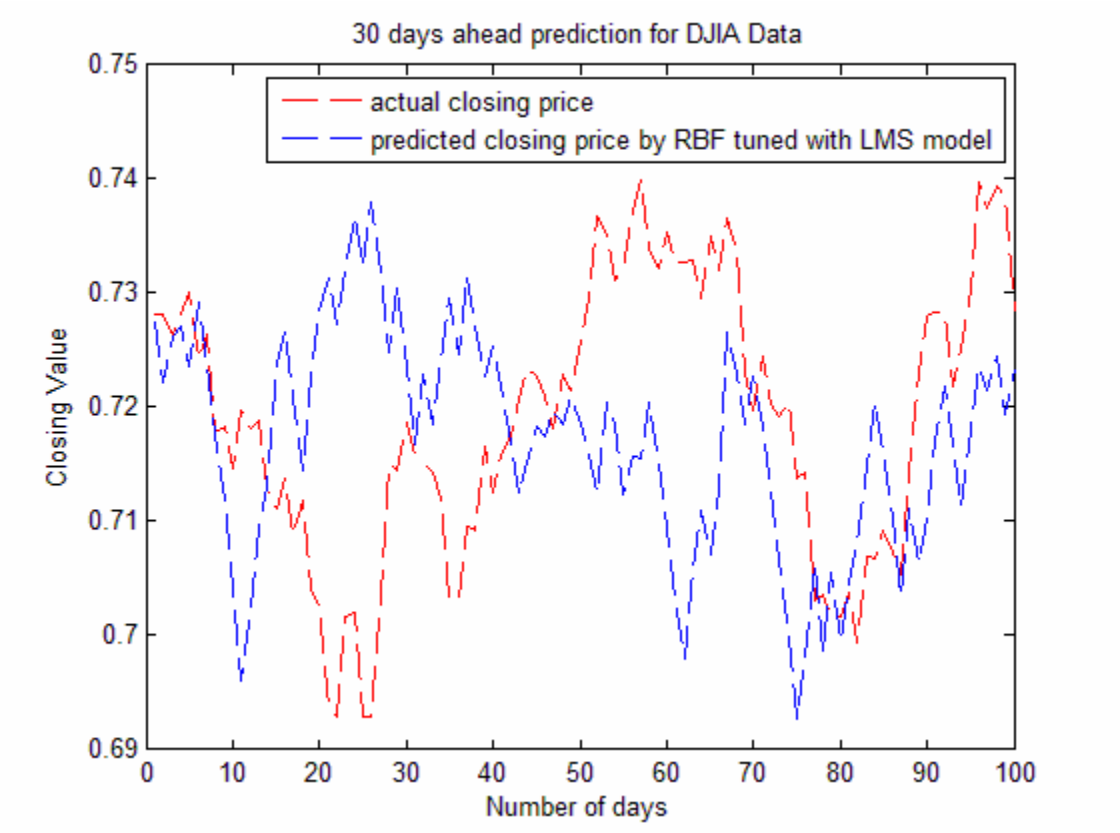

## For 5 days ahead prediction



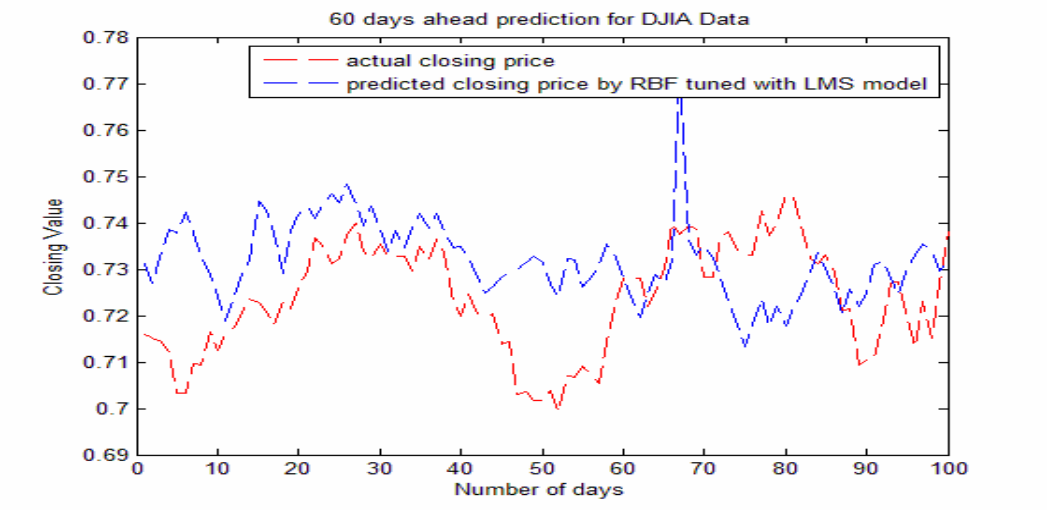
# For 10 days ahead prediction




# For 30 days ahead prediction



## For 60 days ahead prediction




## Model Comparison Via MAPE for DJIA

| Ahead Prediction | RBF Parameters tuned by LMS MAPE (%) | RBF Parameters tuned by PSO MAPE (%) | FLANN Parameters tuned by PSO MAPE (%) | Training Done for |
|---|---|---|---|---|
| 1 day | 3.9936 | 2.649 | **2.192** | 1000 days |
| 5 days | 5.3564 | 3.988 | **3.225** | 1000 days |
| 10 days | 6.1574 | **4.133** | 4.536 | 1000 days |
| | 5.9367 | 3.654 | **3.213** | 2000 days |
| 30 days | 6.783 | **5.255** | 7.322 | 1000 days |
| | 6.345 | **5.045** | 6.754 | 2000 days |
| 60 days | 7.364 | **6.133** | 8.221 | 1000 days |
| | 6.124 | **5.966** | 7.567 | 2000 days |

## 6.2 Simulation Results for BSE

Out of around 2000 days data of BSE, it was found that only 1000 days data is sufficent enough to train the various models for 1 day, 5 day, and 10 days ahead prediction.

For 30 and 60 day ahead prediction, upto 2000 days data is used to train the network.

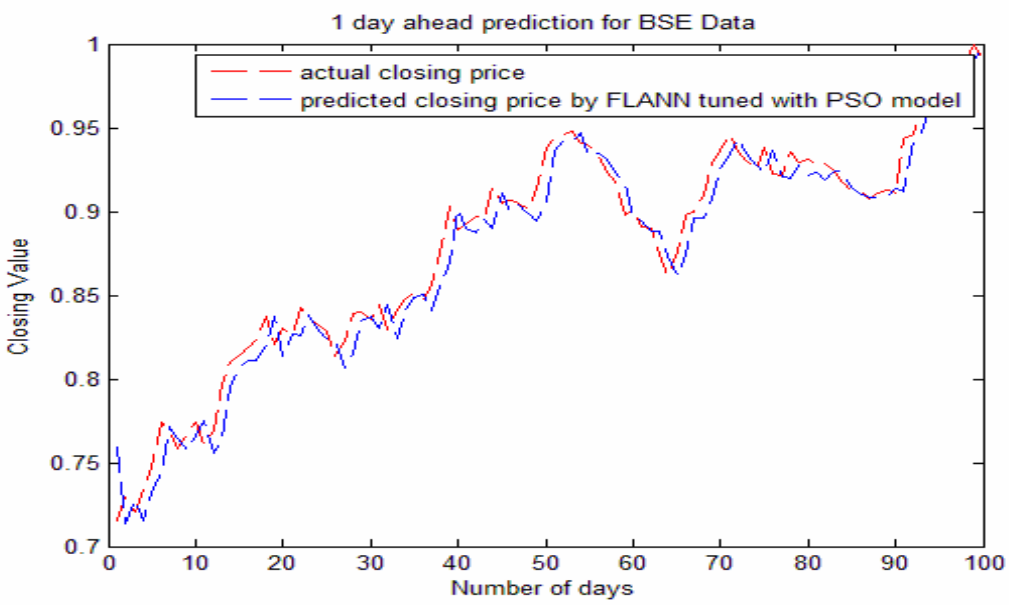
Model is tested with fresh 600 days data , out of which only 100 are shown for clarity.

Tuning Parameters and Associated Parameters:

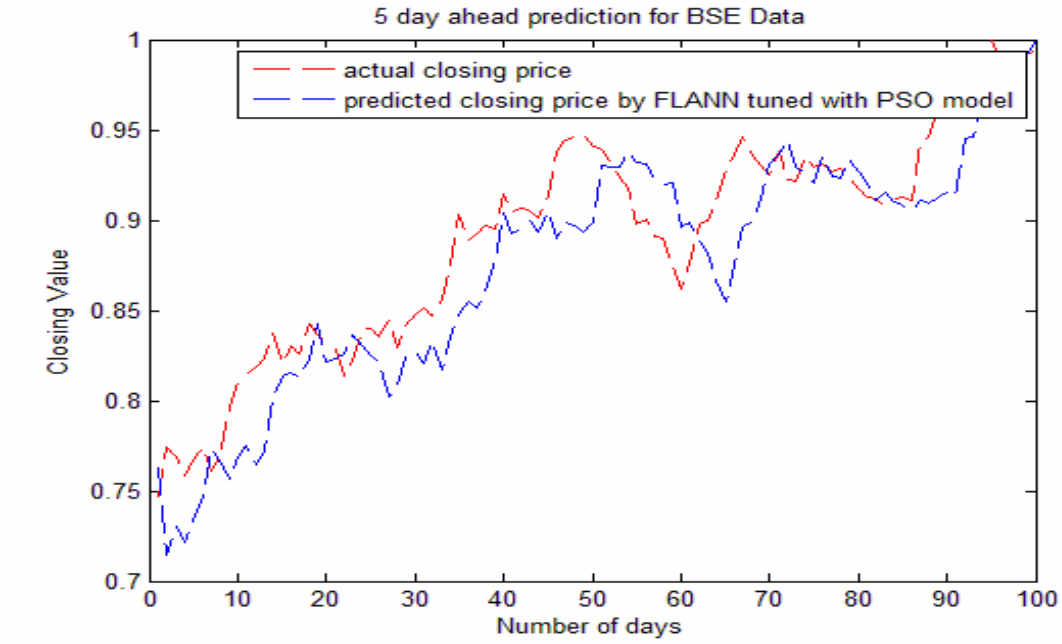
Particles = 20
μ(weight) = 0.4
μ(mean) = 0.5
μ(sig) = 0.7

### 6.2.1 Testing of FLANN Parameters tuned with PSO model

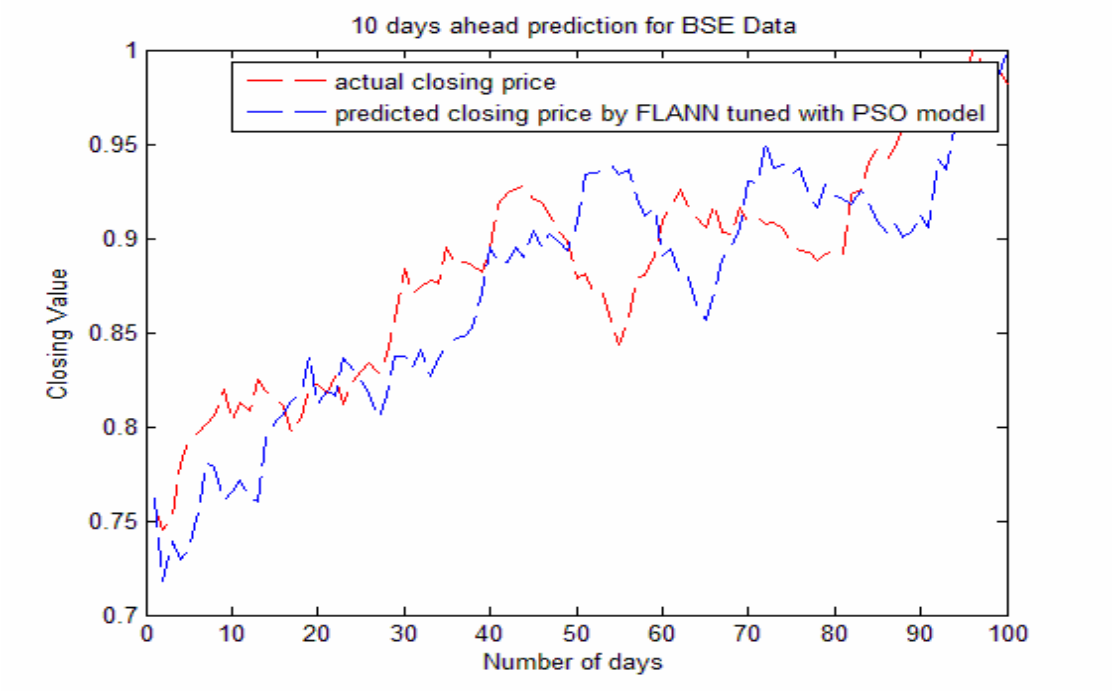
**For 1 day ahead prediction**



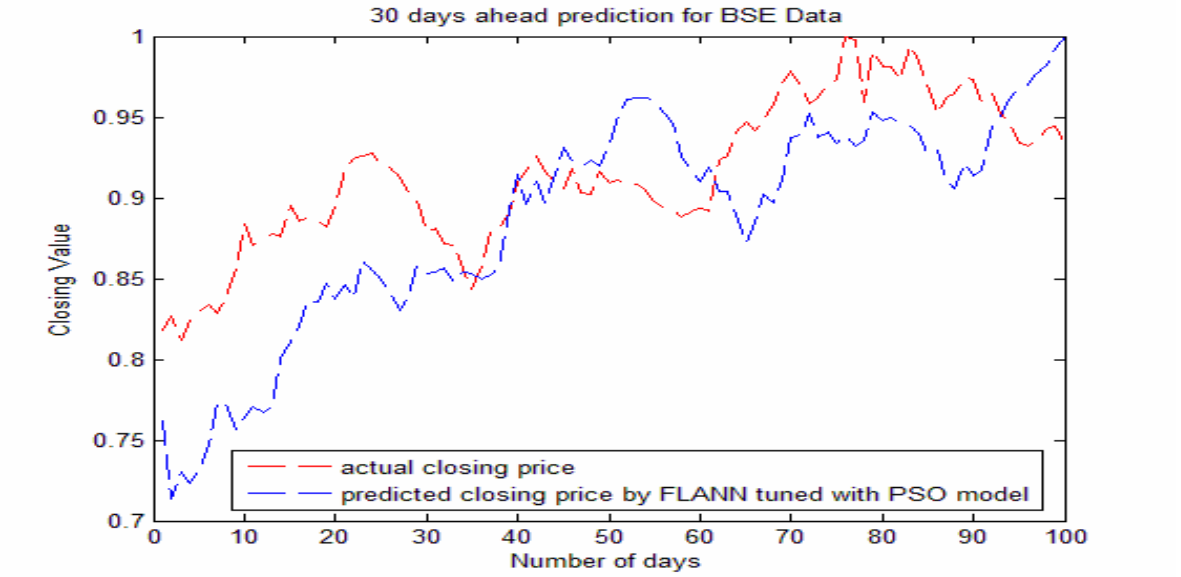

**For 5 days ahead prediction**



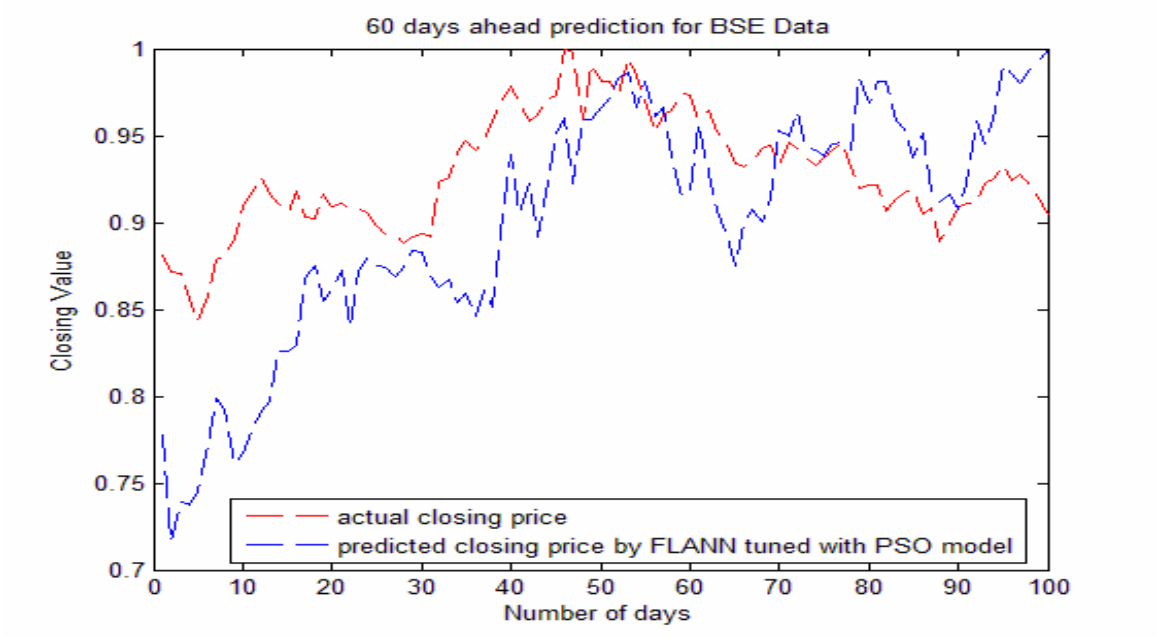
# For 10 days ahead prediction




# For 30 days ahead prediction

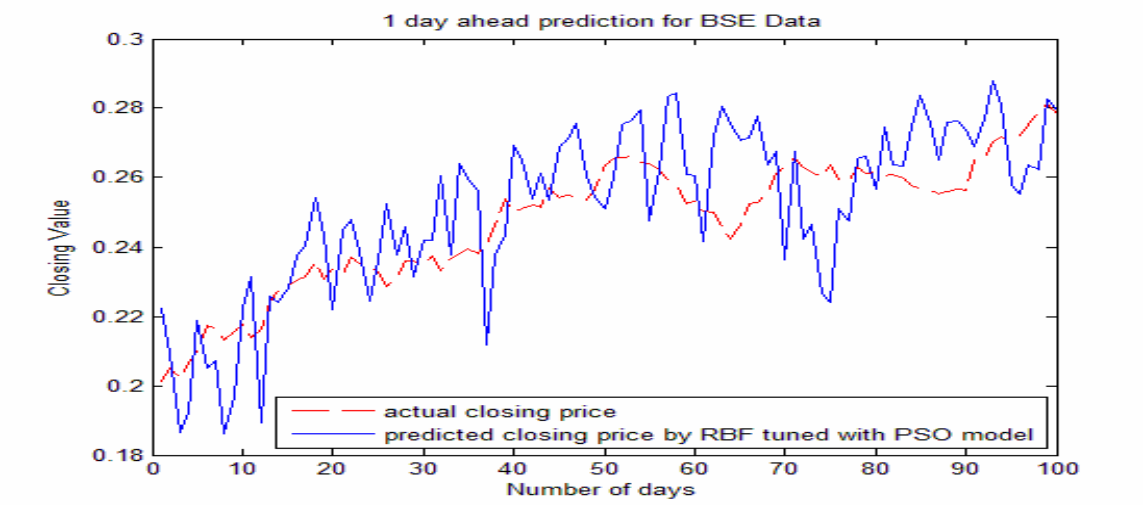


# For 60 days ahead prediction

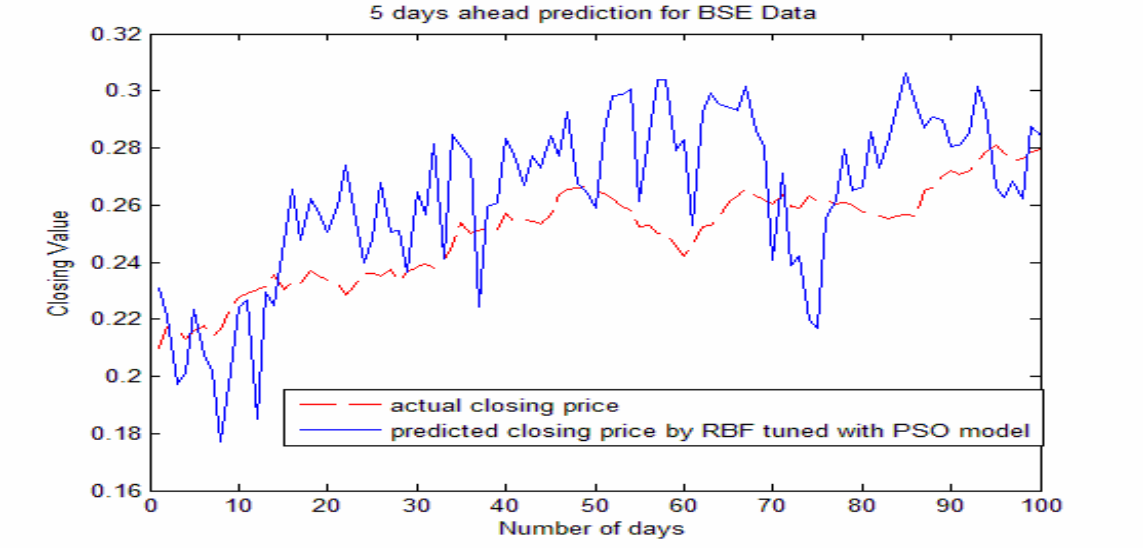


### 6.2.2 Testing of RBF Parameters tuned with PSO model

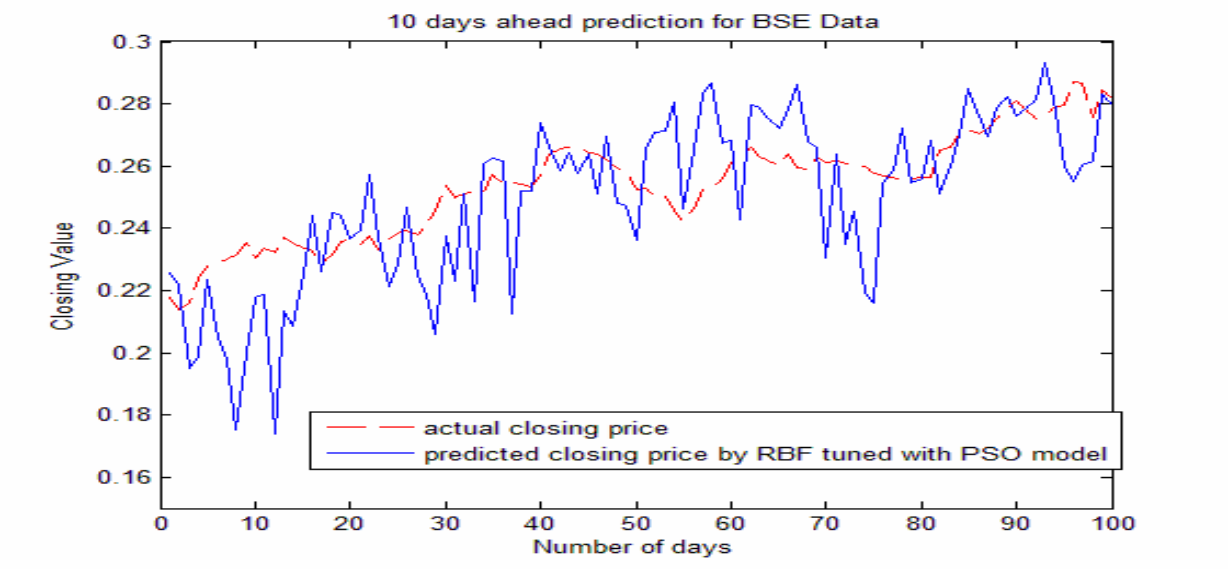
## For 1 day ahead prediction



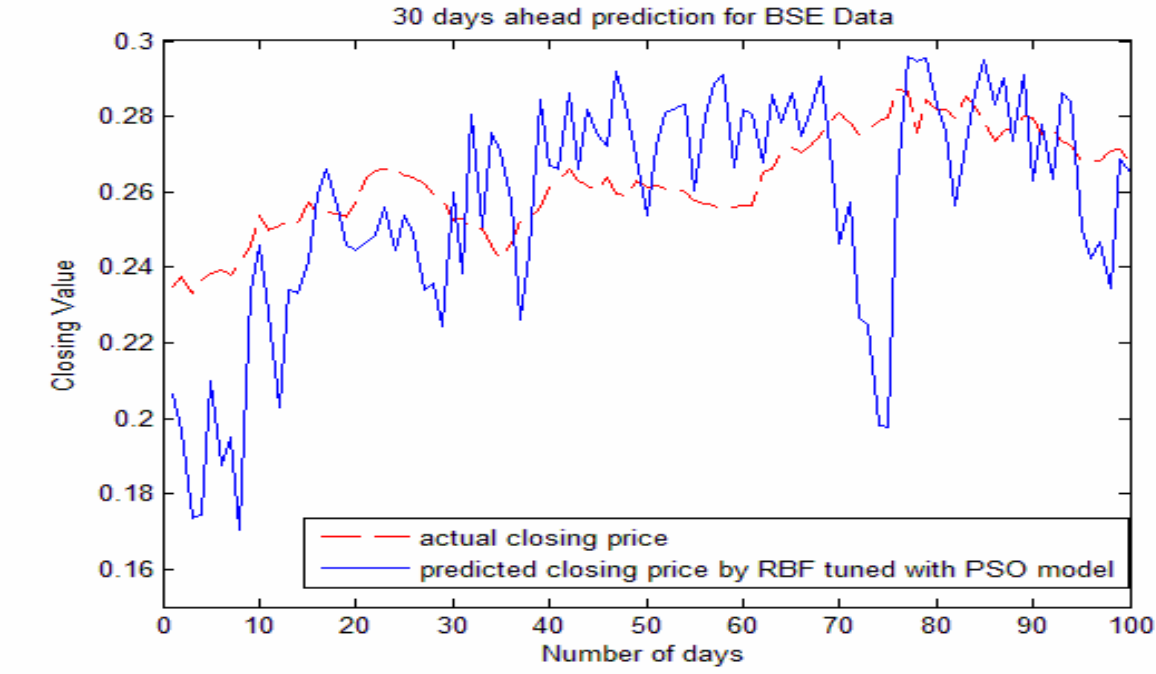

## For 5 days ahead prediction



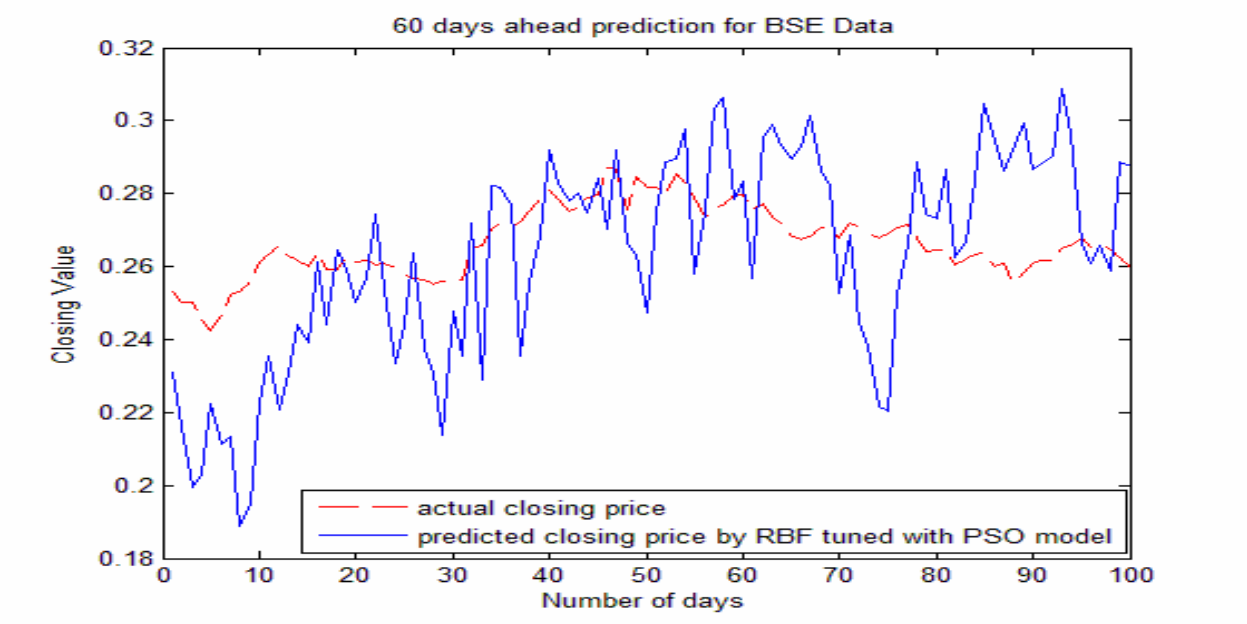
# For 10 days ahead prediction




# For 30 days ahead prediction

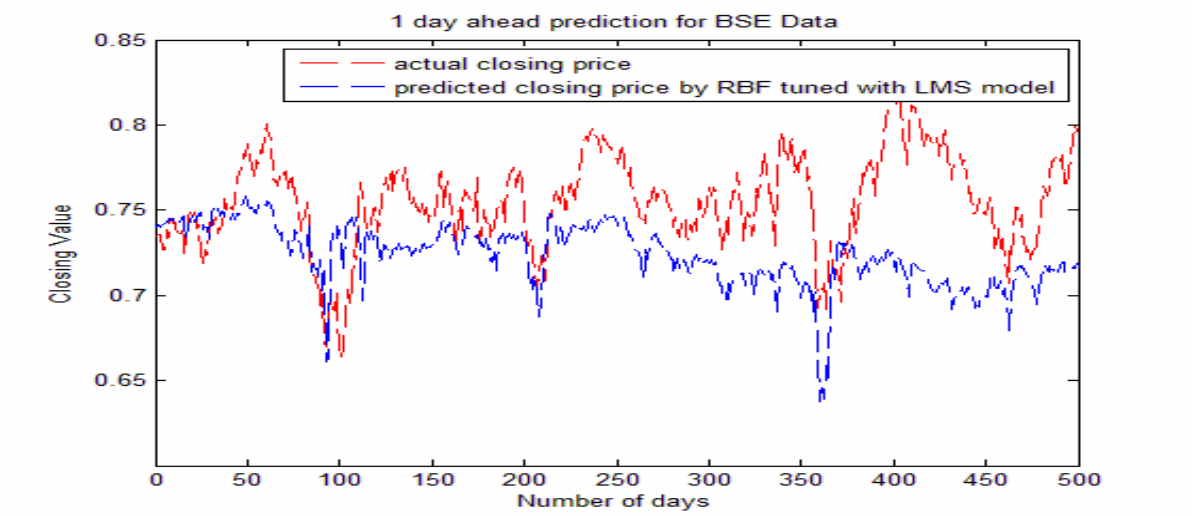


# For 60 days ahead prediction

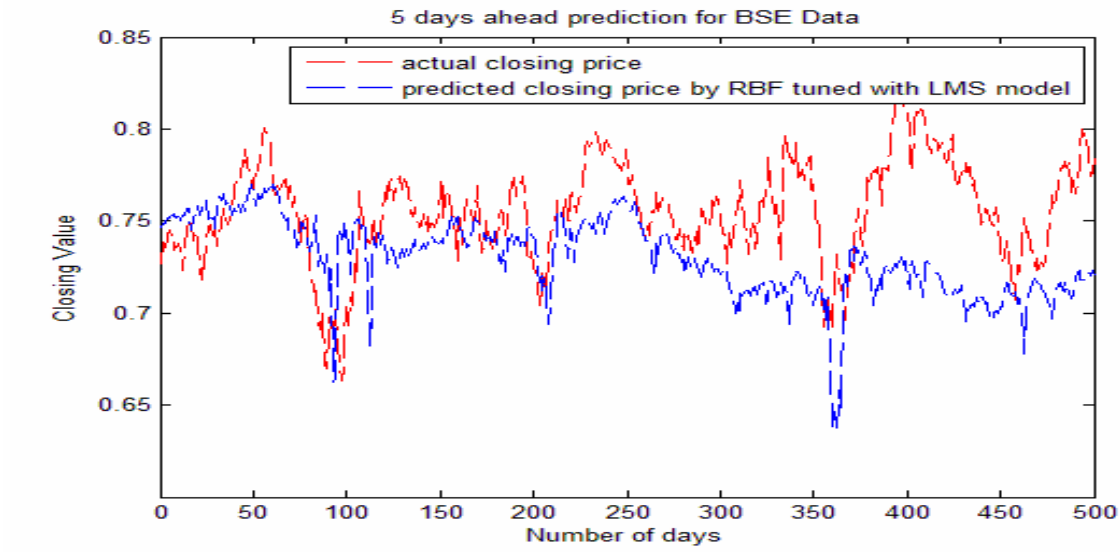


### 6.2.3 Testing of RBF Parameters tuned with LMS model

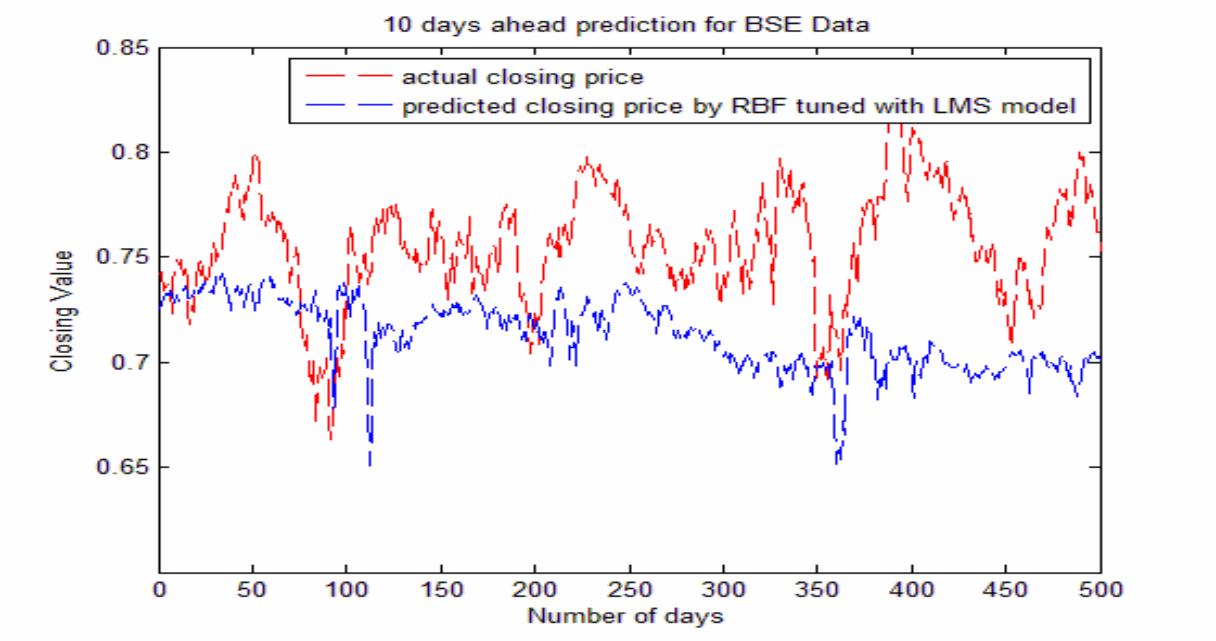
**For 1 day ahead prediction**



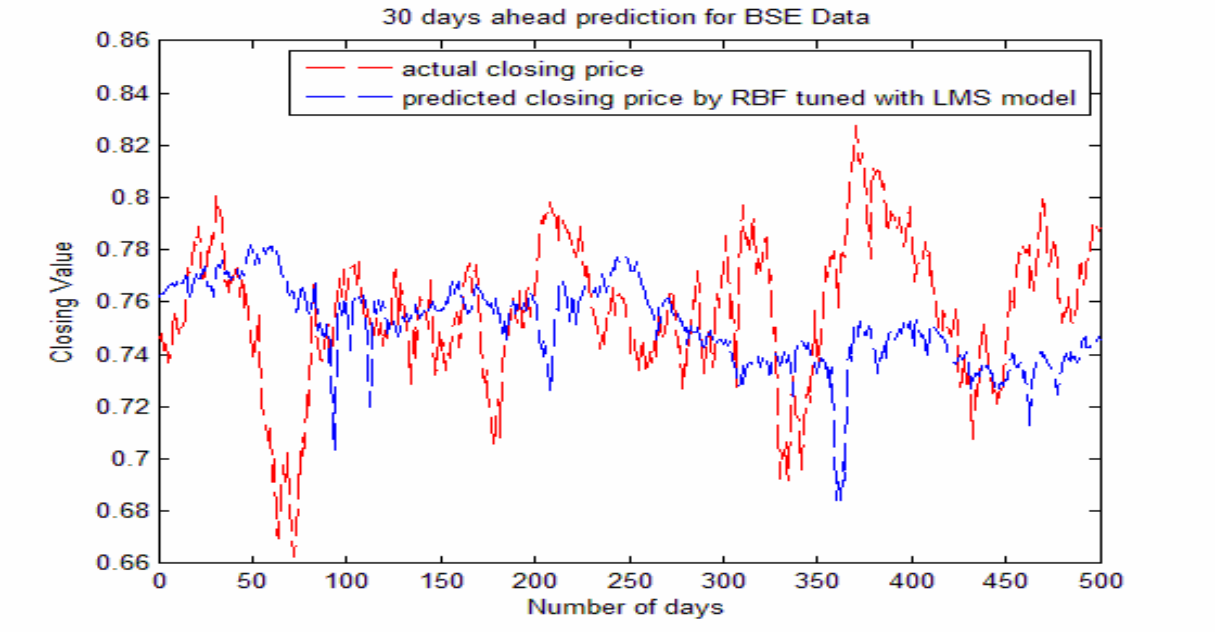

**For 5 days ahead prediction**



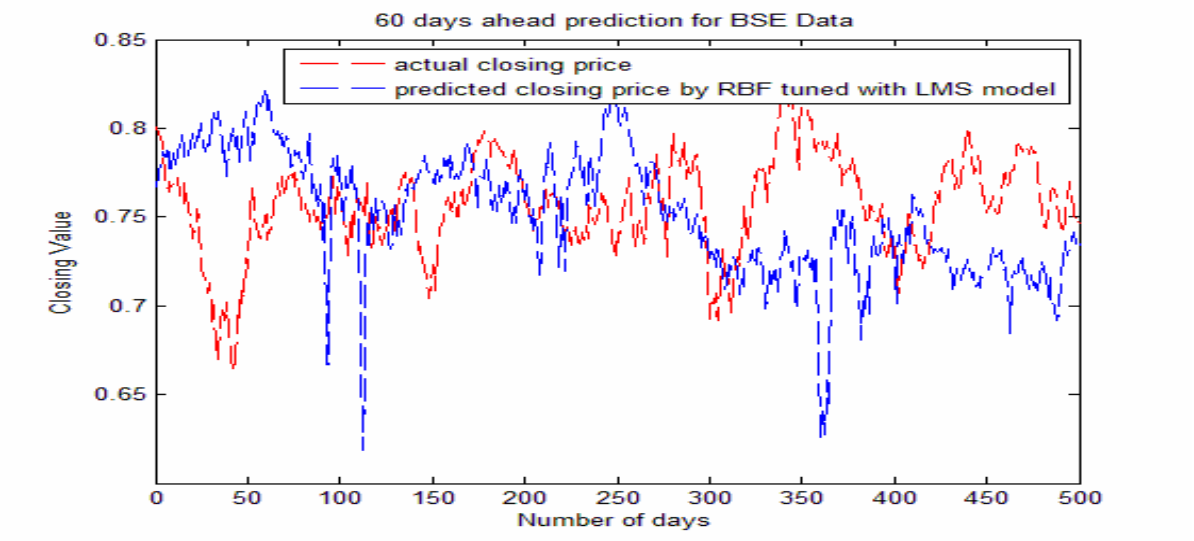
# For 10 days ahead prediction




# For 30 days ahead prediction



## For 60 days ahead prediction




## Model Comparison Via MAPE for BSE

| Ahead Prediction | RBF Parameters tuned by LMS MAPE (%) | RBF Parameters tuned by PSO MAPE (%) | FLANN Parameters tuned by PSO MAPE (%) | Training Done for |
|---|---|---|---|---|
| 1 day | 4.985 | 4.618 | **1.008** | 1000 days |
| 5 days | 4.423 | 5.386 | **2.533** | 1000 days |
| 10 days | 7.677 | 5.967 | **3.867** | 1000 days |
| | 6.047 | 5.935 | **3.611** | 2000 days |
| 30 days | 11.67 | **6.990** | 7.577 | 1000 days |
| | 10.45 | **6.567** | 6.869 | 2000 days |
| 60 days | 13.56 | **7.556** | 9.456 | 1000 days |
| | 12.45 | **7.256** | 8.517 | 2000 days |

## 6.3 Simulation Results for S&P500

Out of around 2000 days data of S&P 500, it was found that only 1000 days data is sufficent enough to train the various models for 1 day, 5 day, and 10 days ahead prediction.

For 30 and 60 day ahead prediction, upto 2000 days data is used to train the network.

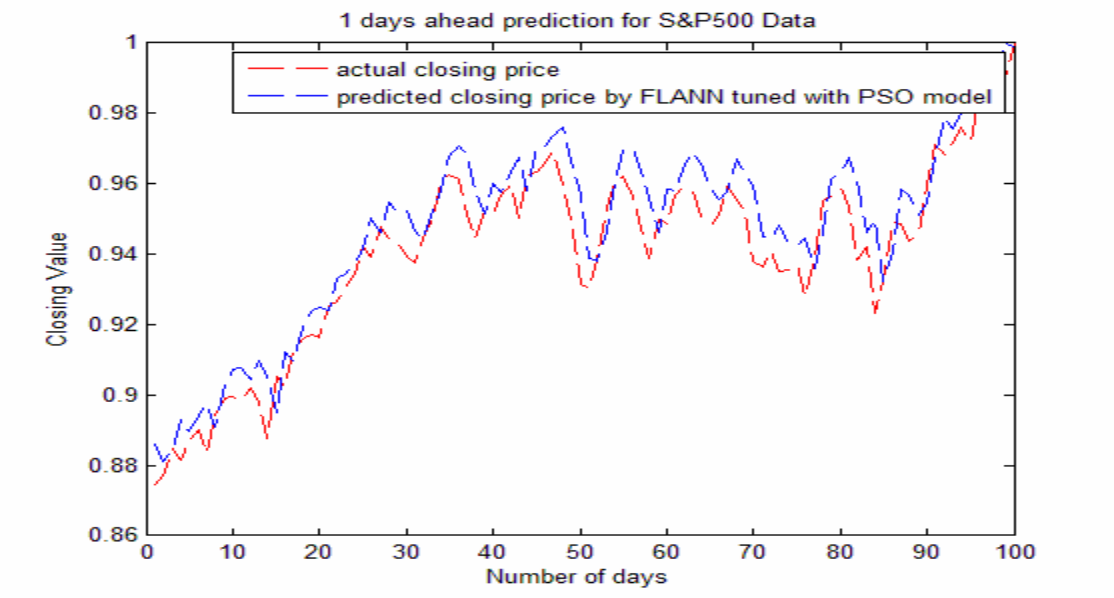
Model is tested with fresh 600 days data , out of which only 100 are shown for clarity.

Tuning Parameters and Associated Parameters:

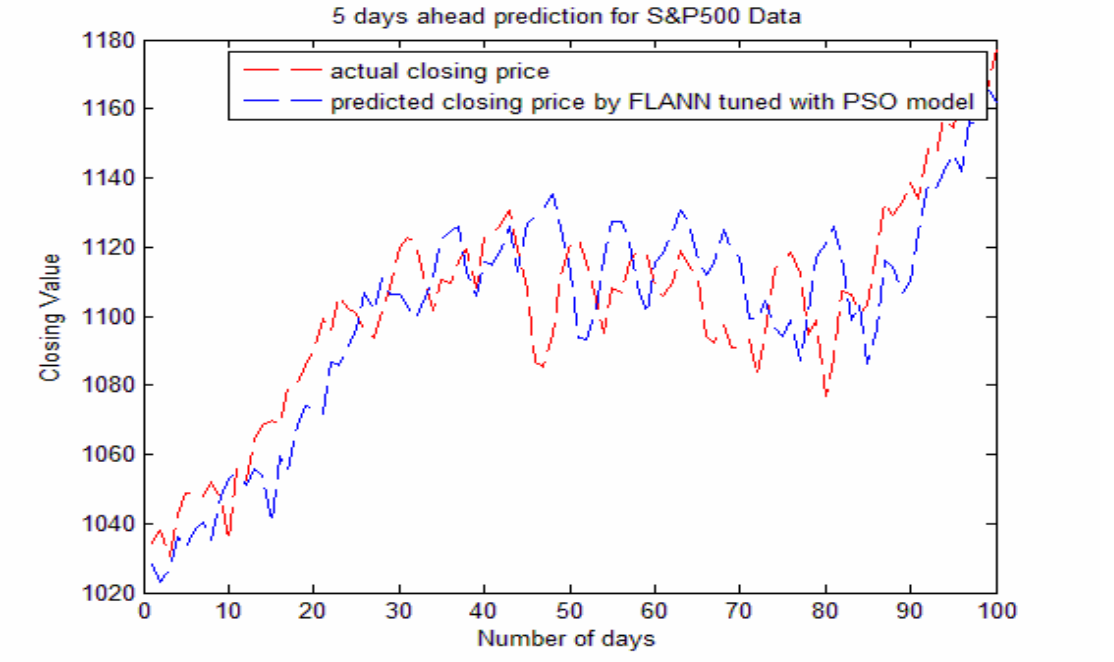
Particles = 20
μ(weight) = 0.4
μ(mean) = 0.5
μ(sig) = 0.7

### 6.3.1 Testing of FLANN Parameters tuned with PSO model

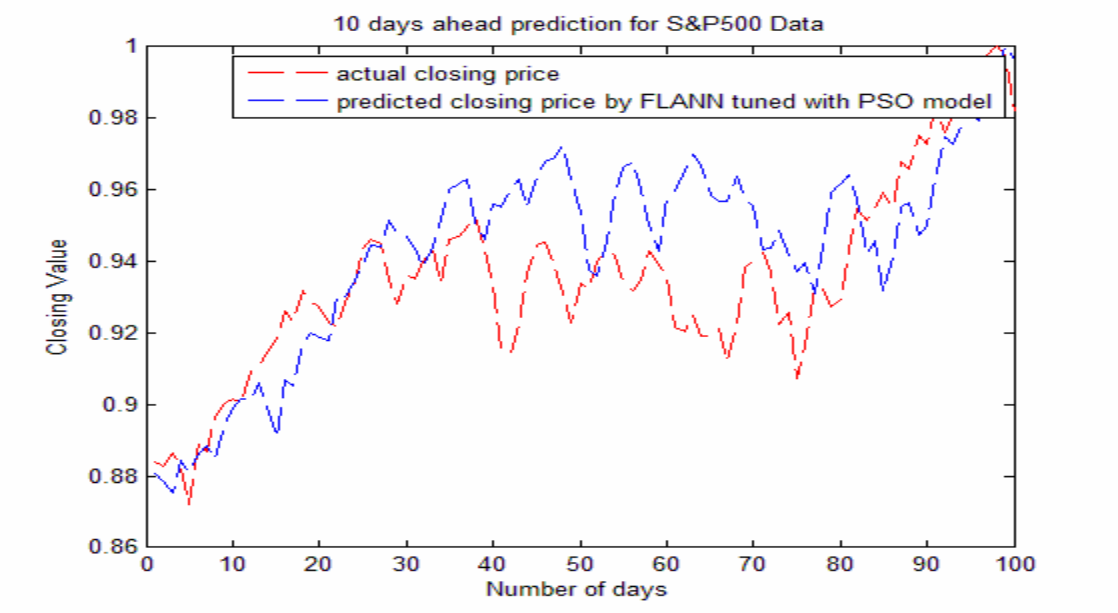
## For 1 day ahead prediction



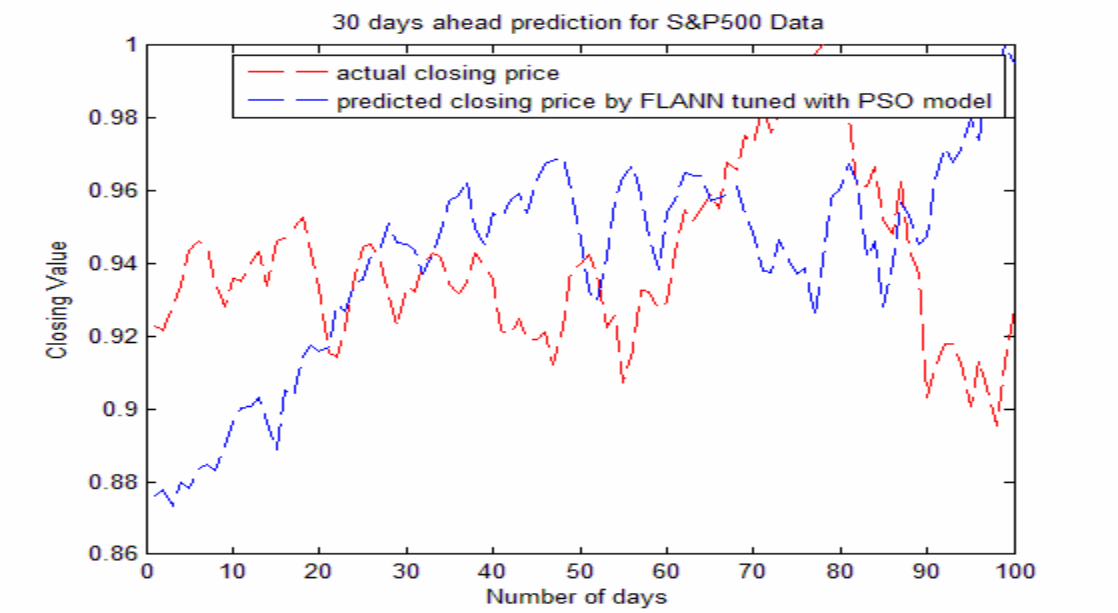

## For 5 days ahead prediction



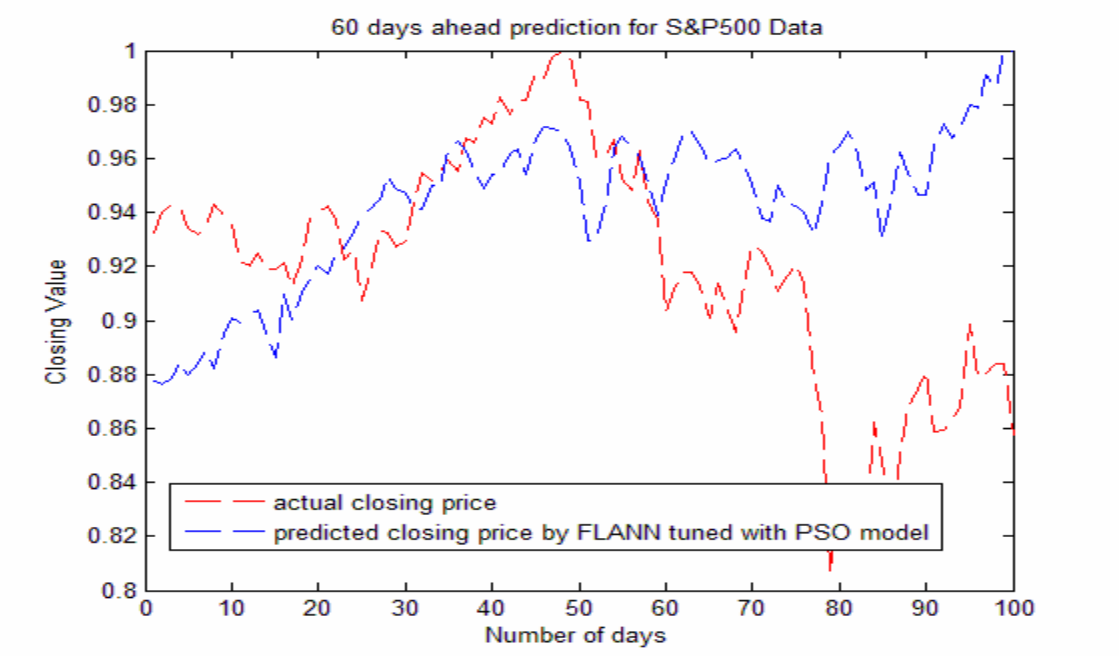
# For 10 days ahead prediction




# For 30 days ahead prediction

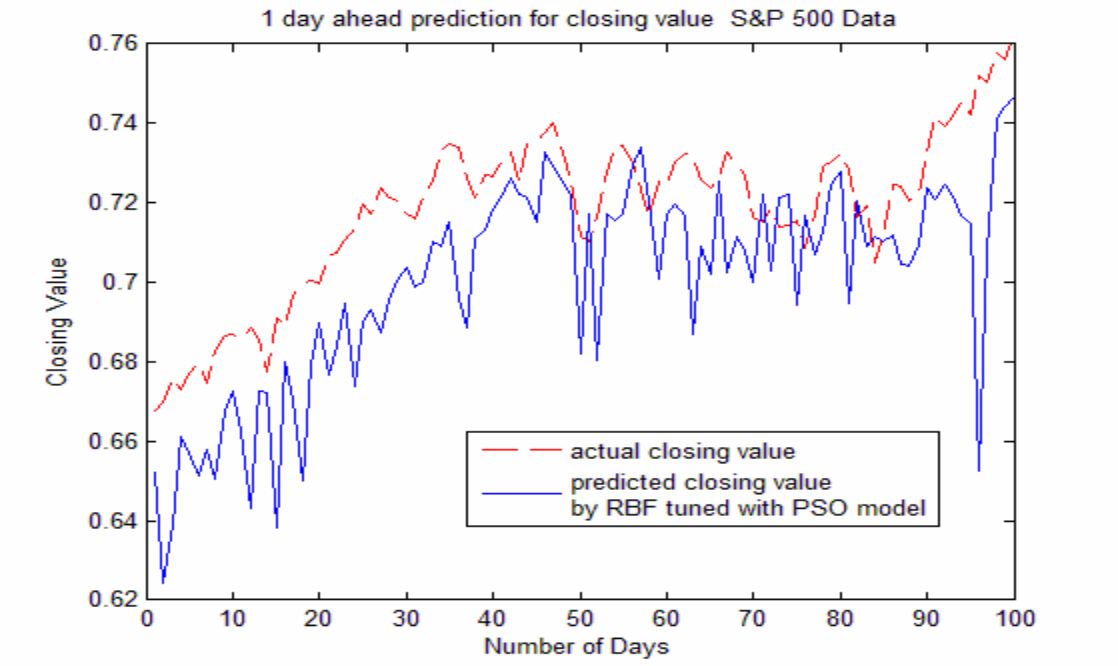


# For 60 days ahead prediction

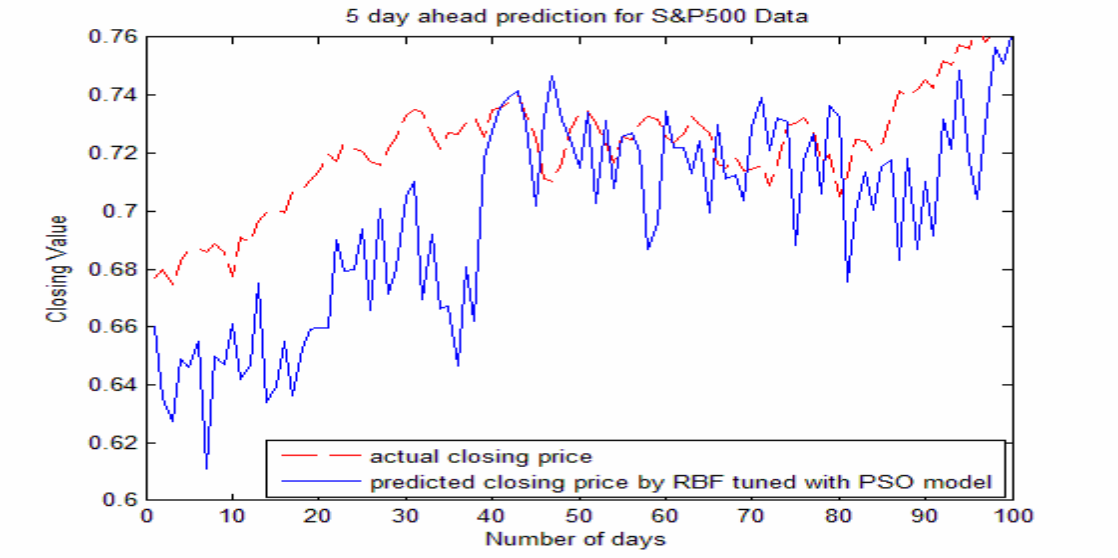


### 6.3.2 Testing of RBF Parameters tuned with PSO model

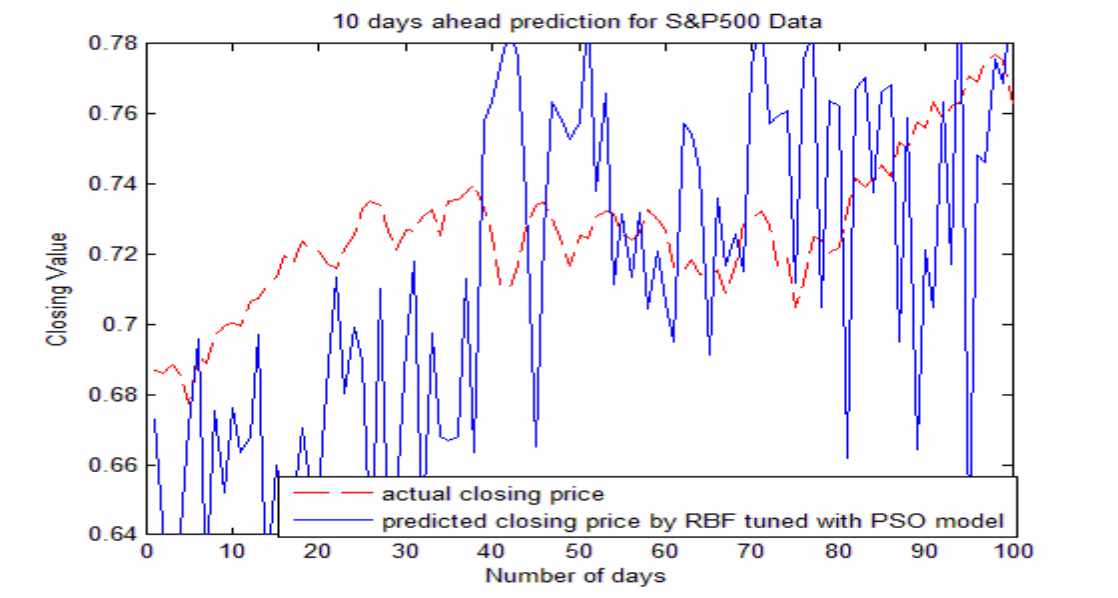
## For 1 day ahead prediction



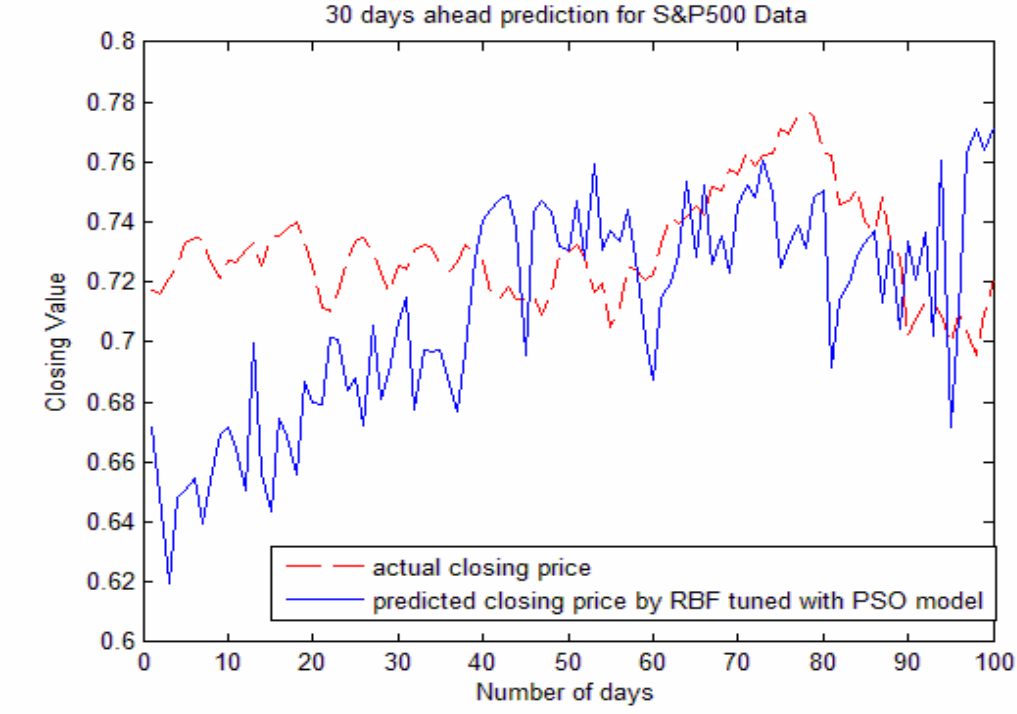

## For 5 days ahead prediction



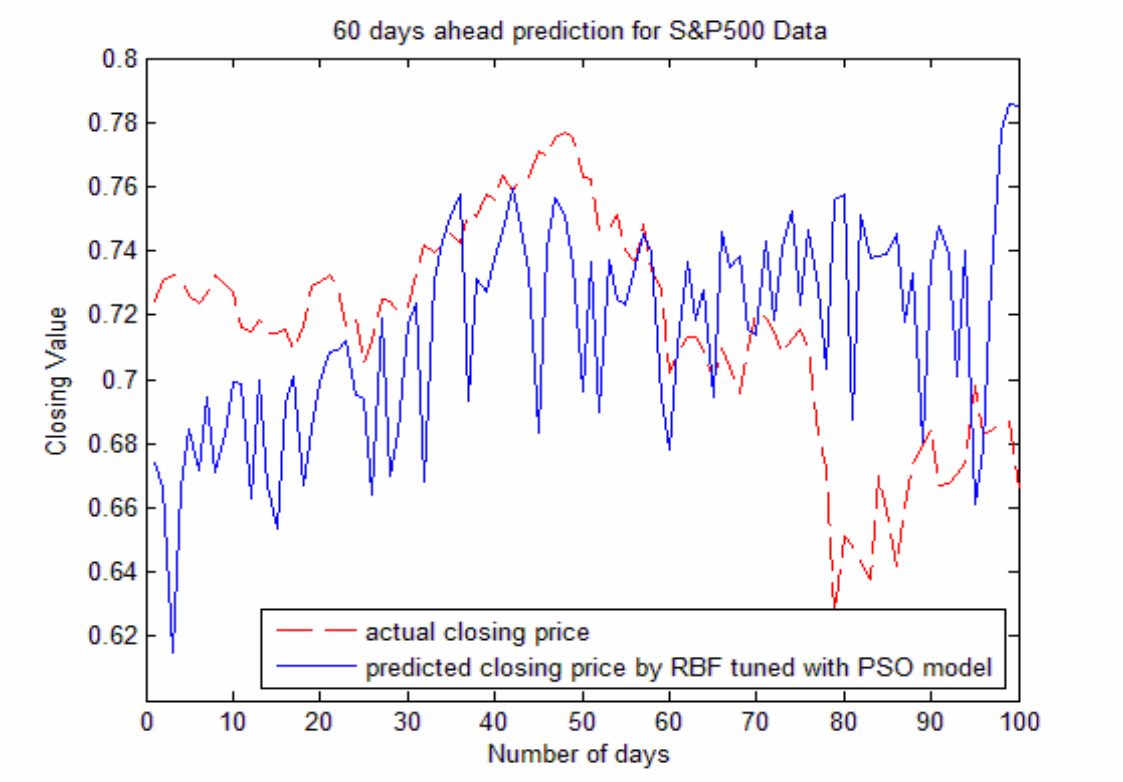
# For 10 days ahead prediction




# For 30 days ahead prediction

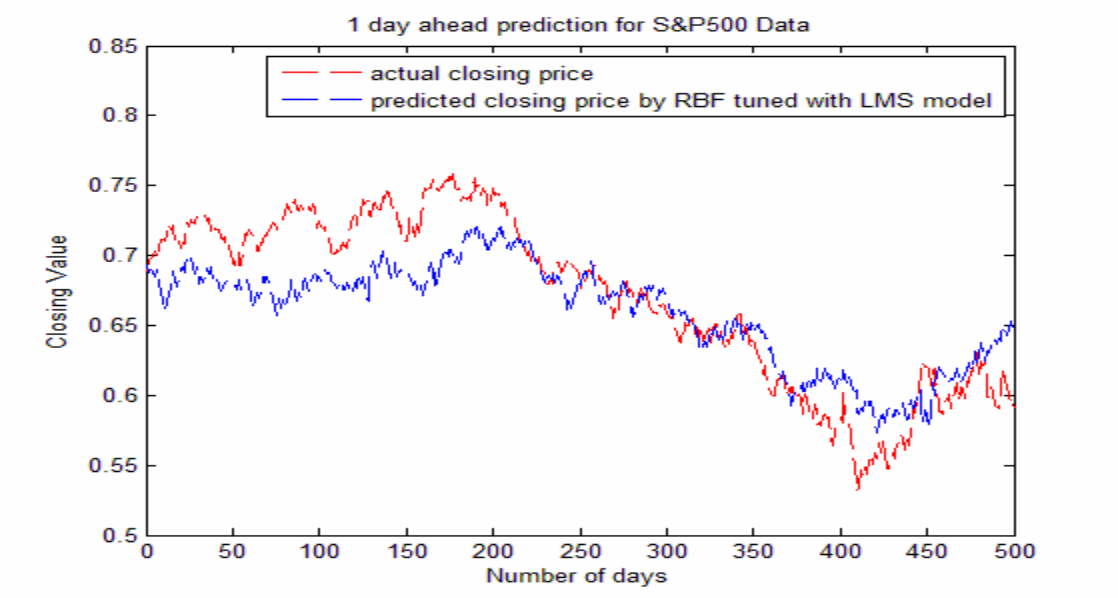


# For 60 days ahead prediction

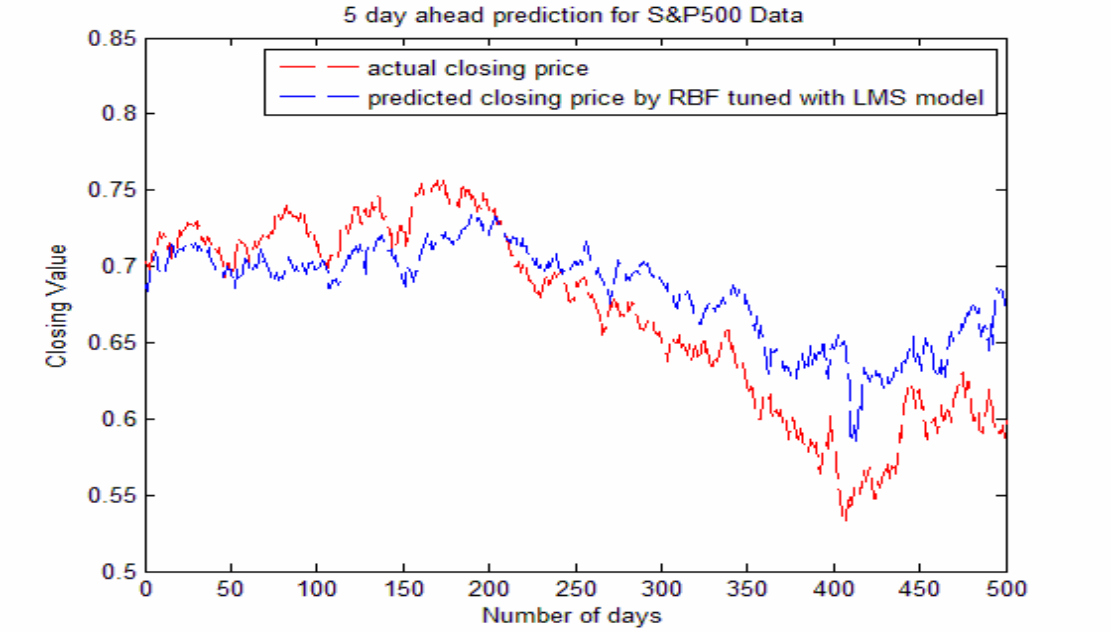


### 6.3.3 Testing of RBF Parameters tuned with LMS model

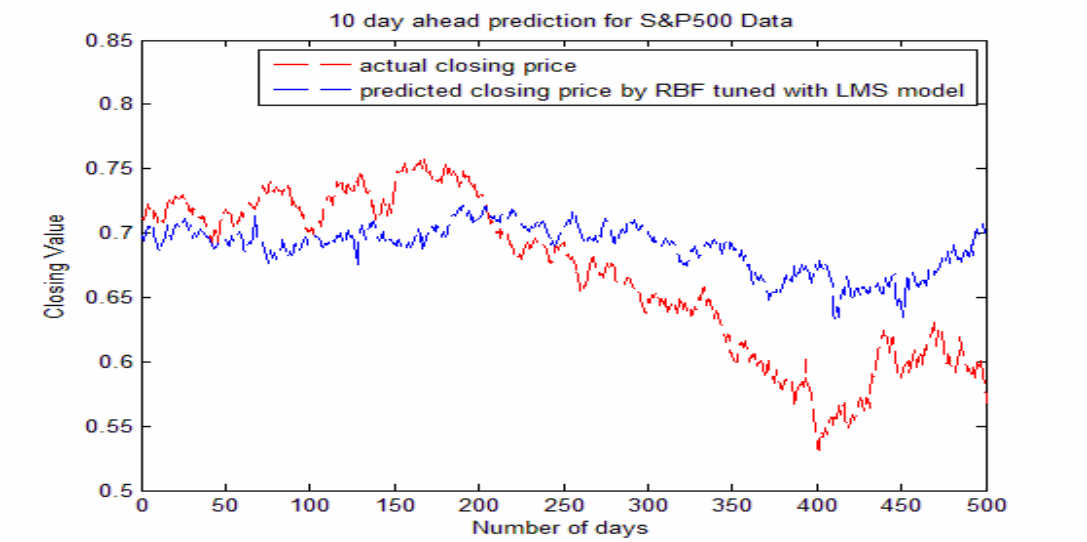
## For 1 day ahead prediction



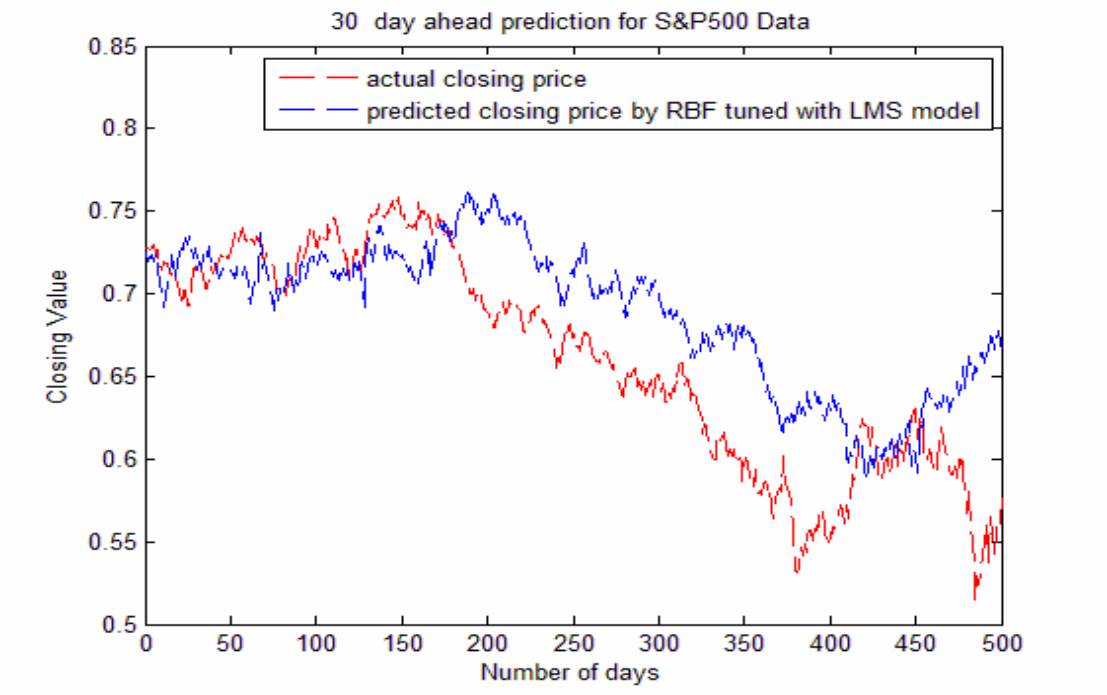

## For 5 days ahead prediction



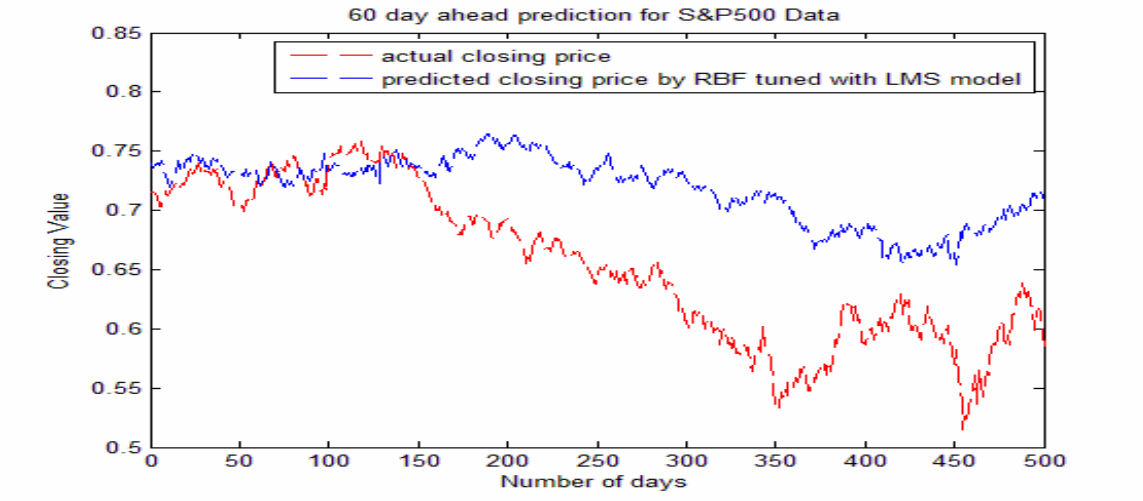
# For 10 days ahead prediction




# For 30 days ahead prediction



## For 60 days ahead prediction




## Model Comparison Via MAPE for S&P 500

| Ahead Prediction | RBF Parameters tuned by LMS MAPE (%) | RBF Parameters tuned by PSO MAPE (%) | FLANN Parameters tuned by PSO MAPE (%) | Training Done for |
|---|---|---|---|---|
| 1 day | 1.2503 | 1.194 | **0.987** | 1000 days |
| 5 days | 3.7390 | 3.986 | **2.196** | 1000 days |
| 10 days | 5.874 | 6.438 | **3.571** | 1000 days |
| | 4.864 | 5.438 | **2.773** | 2000 days |
| 30 days | 6.156 | **5.490** | 6.789 | 1000 days |
| | 5.877 | **4.434** | 5.233 | 2000 days |
| 60 days | 9.4456 | **6.220** | 7.767 | 1000 days |
| | 8.8156 | **5.220** | 6.154 | 2500 days |

# Chapter 7

# DISCUSSION & CONCLUSION

## 7.1 Discussion

Starting from the calculation of various parameters associated with the Stock exchange data, in this thesis we have formulated a comparision between various learning models like PSO and LMS. LMS has the advantage of faster convergence but the probability of getting stuck in local optima is high. On the other hand, PSO is a derivative free technique but slower compared to LMS. Comparison of both methods are done by taking RBF and FLANN models and results are compiled.

In our lucid simulations, we tested our algorithms on three types of stock exchanges namely DJIA, S&P500 and BSE. The results are compiled along with the response plots.

## 7.2 Conclusion

The Functional Link Artificial Neural Network and Radial basis function based stock market prediction model is introduced. With the use of FLANN and RBF, the model for prediction of stock market indices becomes simpler and involves lesser computations compared to other such model reported earlier. Experiments show that in case of lower number of ahead prediction, FLANN parameters updated with PSO algorithm gives the best result. While for higher number of days ahead predicition, RBF parameters updated with PSO algorithm works the best.

# Chapter 8

## REFERENCES

# REFERENCES


[1]. T.Z. Tan, C. Quck, G.S. Nag, "Brain Inspired Genetic Complimentary Learningfor Stock Market Prediction"

[2]. K.J. Oh, K. Kim, -j (2002) "Analyzing stock market tick data using piecewise non linear model." Expert System with Applications, 22, P 249-255.

[3]. Y. Wang, -F(2003) "Mining stock prices using fuzzy rough set system" Expert System with Applications,24, P13-23.

[4]. Y. Wang –F(2002) "Predicting stock price using fuzzy grey prediction system" Expert System with Applications,22, P33-39.

[5]. E. Peters, "Chaos & Order in Capital Markets: A new view of cycles, prices, market volatility", John Wiley & Sons Inc.

[6]. S. Taylor, "Modeling Financial Time Series", John Wiley & Sons (1986).

[7]. T. Masters, "Practical Neural Network recipes in C++", Academic Press, NewYork, 1993.

[8]. J. Hanm, N. Kamber (2001), " Data Mining: Concepts & Techniques, San Francisco; Morgan Kanfmann Publishers

[9]. E.W. Saad , D.V. Prokhorov, D.C. Wunsch (1998), " Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks", IEEE Transactions of Neural Network, 9(6), 1456-1470.

[10]. Z. Chen (2000), "Computational Intelligence for Decision Support", Florida,USA; CRC Press

[11]. Y-H. Pao, "Adaptive Pattern Recognition & Neural Networks", Reading, MA; Addison-Wesley, 1989.

[12]. Y-H. Pao, S.M. Phillips, D.J. Sobajic," Neural Net Computing and intelligent control systems", Int. J. Contr. Vol. 56, No. 2, P 263-289, 1992.

[13]. J.C. Patra, R.N. Pal, B.N. Chatterji, G. Panda, "Identification of non-linear & dynamic system using functional link artificial neural network ", IEEE Transactions on System, Man & Cybernetics – Part B; Cybernetics, Vol. 29, No. 2, April 1999.

[14]. Kimoto,T., Asakawa K., Yoda M., Takeoka M., "Stock market prediction system with modular neural networks", IJCNN International Joint Conference on Neural Networks, 1990., 1990 17-21 June 1990 Page(s):1 - 6 vol.1

[15]. Clarence N.W. Tan and Gerhard E. Wittig, "A Study of the Parameters of a Backpropagation Stock Price Prediction Model", Proceedings 1993 The First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems p. 288-91,1993

[16]. Tan, H.; Prokhorov, D.V.; Wunsch, D.C., II; "Conservative thirty calendar day stock prediction using a probabilistic neural network", Computational Intelligence for Financial Engineering, 1995., Proceedings of the IEEE/IAFE 1995 9-11 April 1995 Page(s):113 – 117

[17]. Ornes, C.; Sklansky, J.; "A neural network that explains as well as predicts financial market behavior", Computational Intelligence for Financial Engineering 1997, Proceedings of the IEEE/IAFE 1997 24-25 March 1997 Page: 43 – 49.

[18]. Yamashita, T.; Hirasawa, K.; Jinglu Hu; "Application of multi-branch neural networks to stock market prediction", IEEE International Joint Conference on Neural Networks, 2005. IJCNN '05. Volume 4, Aug. 2005 Page(s):2544 – 2548 vol. 4.

[19]. Yuehui Chen; Xiaohui Dong; Yaou Zhao; "Stock Index Modeling using EDA based Local Linear Wavelet Neural Network", International Conference on Neural Networks and Brain, 2005. Vol. 3, 13-15 Oct. 2005 Page(s):1646 –1650

[20]. Lee, R.S.T.; " iJADE stock advisor: an intelligent agent based stock prediction system using hybrid RBF recurrent network" , IEEE Transactions on Systems, Man and Cybernetics, Part A, Volume 34, Issue 3, May 2004 Page(s):421 – 428.

[21]. Ray Tsaih, Yenshan Hsu, Charles C. Lai; "Forecasting S&P 500 stock index futures with a hybrid AI system", Decision Support Systems 23 1998. pages: 161–174.

[22]. Hiemstra, Y.; "A stock market forecasting support system based on fuzzy logic", Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, 1994. Vol. III: Information Systems: Decision Support and Knowledge-Based Systems, Volume 3, 4-7 Jan. 1994 Page(s):281 – 287.

[23]. Sheta, A.; "Software Effort Estimation and Stock Market Prediction Using Takagi-Sugeno Fuzzy Models" , IEEE International Conference on Fuzzy Systems, 2006 July 16-21, 2006 Page(s):171 – 178.

[24]. Badawy, F.A.; Abdelazim, H.Y.; Darwish, M.G.; "Genetic Algorithms for Predicting the Egyptian Stock Market", 3rd International Conference on Information and Communications Technology, 2005. Dec. 2005. P109 – 122

[25]. Tan, T.Z.; Quek, C.; Ng, G.S.; "Brain-inspired genetic complementary learning for stock market prediction", IEEE Congress on Evolutionary Computation, 2005. Volume 3, 2-5 Sept. 2005 Page(s):2653 - 2660 Vol. 3.

[26]. Kyoung-jae Kim; “Artificial neural networks with evolutionary instance selection for financial forecasting”, Expert Systems with Applications 30, 2006 pages 519-526.

[27]. R.J. Kuo; C.H. Chen, Y.C. Hwang; “An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network”, Fuzzy Sets and Systems 118 (2001) pages 21-45.

[28]. Md. Rafiul Hassan , Baikunth Nath, Michael Kirley; “A fusion model of HMM, ANN and GA for stock market forecasting” , Expert Systems with Applications 2006.

[29]. Hyun-jung Kim, Kyung-shik Shin, “A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets” , Applied Soft Computing 2006, March 2006.